CORAZONISTAS BARCELONA

Olimpiada 2016

Trabajo de Final de Grado

Javier Gárate Vidiella

Plataforma de gestión. Olimpiada Corazonista 2016

Titulación: Grado en Ingeniería Informática. Autor: Javier Gárate Vidiella. Directora: Sílvia Llorente Viejo.
Centro: Facultad de Informática de Barcelona. Universidad Politécnica de Cataluña.

Agradecimientos

Me gustaría dar las gracias a todas las personas que, de una manera u otra, han hecho posible la realización de este TFG:

A Sílvia Llorente, tutora de este proyecto, por darme libertad absoluta tanto en la temática como en las tecnologías a utilizar, y por su ayuda a la hora de documentar.

Al AMPA del colegio Corazonistas Barcelona, por su colaboración y su dedicación en todas las etapas del proyecto.

Al equipo de CIMNE, por su apoyo y ayuda durante el desarrollo. En especial a Sergio Valero, por su paciencia inacabable, por enseñarme las sendas de la programación en .NET, sus consejos y sus ganas.

En general, a todos los que han colaborado en hacerme llegar hasta aquí.

Gracias a todos.

Abstract

Castellano

En este documento se explica el desarrollo de un sistema web para la gestión de un evento deportivo, basado en el framework de Microsoft .NET. Este sistema consiste en 2 webs, una de acceso público y una intranet de gestión. Se explica la planificación inicial, los presupuestos, los requisitos impuestos por el cliente, así como muestras de la especificación, el diseño y la implementación.

Este sistema web fue utilizado para la gestión de un evento multideportivo de unos 23 equipos y 272 participantes, celebrado en Barcelona los días 6, 7 y 8 de mayo de 2016.

Català

En aquest document s'explica el desenvolupament d'un sistema web per a la gestió d'un esdeveniment esportiu, basat en el framework de Microsoft .NET. Aquest sistema consisteix en 2 webs, una d'accés públic i una intranet de gestió. S'expliquen la planificació inicial, els pressupostos, els requisits impulsats per la organització, així com alguns fragments de la especificació, el disseny i la implementació.

Aquest sistema web es va utilitzar per a la gestió d'un esdeveniment multiesportiu, que va acollir a 23 equips y 272 participants, celebrat a Barcelona els dies 6, 7 i 8 de maig de 2016.

English

This document explains the development of a web distributed system for the management of a sport event, based on Microsoft .NET framework. The system consists of 2 websites, a public one, and an intranet for the management. In this document the following items are explained: initial temporal Schedule, the budget, the requirements imposed by the customer, as well as samples of the specification, design and implementation.

This web system was used for the management of a multi-sport event, with 23 teams and 272 participants disputed in Barcelona on 6, 7 and 8 May 2016

Contenido

Olimpiada Corazonista 2016: Barcelona

1. Introducción

1.1 Definición del proyecto

Este Trabajo de final de grado (TFG) consiste en el diseño e implementación de un sistema web para la gestión de un evento deportivo, la XXXIV Olimpiada Corazonista. Este evento se celebra anualmente entre los 10 colegios Corazonistas de España y alumnos de entre 5º de primaria y 2 de secundaria compiten en baloncesto, balonmano y fútbol. Los días 6, 7 y 8 de mayo de 2016 se celebró en Barcelona, y 272 participantes disfrutaron de un fin de semana de convivencia y deporte.

La convivencia es una parte fundamental del espíritu olímpico. Por eso, durante todo el evento, los participantes que vienen de otros colegios son acogidos por las familias del colegio organizador. Los colegios participantes fueron: Madrid, Moncayo (Zaragoza), La Mina (Zaragoza), Valladolid, Haro (la Rioja), Alsasua, Vitoria, San Sebastián, Rentería (San Sebastián) y Barcelona.

El proyecto constará de una aplicación web de gestión y de una página web de información pública.

1.2 Motivación personal

Tras 8 años participando como entrenador, y 1 como jugador, y haber observado los procesos y formularios en papel que los colegios deben cumplimentar para inscribir a sus equipos y participantes, y la organización para gestionar las acogidas, decidí aportar mi granito de arena y proporcionarles una plataforma web que facilitara las gestiones y reutilizable en las próximas olimpiadas.

1.3 Objetivos

El objetivo principal es ofrecer una versión probada del sistema web a desarrollar a tiempo para la fecha del evento, que satisfaga los siguientes puntos:

- Ofrecer a los usuarios toda la información referente al evento.
- Permitir a los colegios realizar las inscripciones de los equipos y los participantes, así como modificar y eliminar sus datos.
- Permitir a la organización llevar un control de todos los equipos y participantes inscritos, así como realizar modificaciones y sacar listados.
- Permitir a las familias del colegio organizador presentar las solicitudes de acogida.
- Ofrecer información sobre calendarios de partidos y estado de la competición.
- Ofrecer información a tiempo real sobre partidos en juego.
- Ofrecer a los visitantes información general sobre las actividades organizadas para visitar y conocer Barcelona.

1.4 Descripción del proyecto

Este proyecto sirve para la gestión del evento, incluyendo las siguientes funcionalidades:

- Gestión de la competición.
  - Participantes
  - Equipos
  - Partidos (en directo)
  - Clasificaciones
- Gestión de Voluntarios
  - Publicación y registro de todas las plazas y turnos
- Gestión de Acogida
  - Solicitudes
  - Asignaciones
- Gestión multimedia y redes sociales
  - Fotos, videos y concursos
- Extra
  - Información de Barcelona
    - Hoteles por la zona
    - Restaurantes
    - Ocio
  - Meteorología

Para lograr estos objetivos, se planifica el desarrollo de los siguientes sistemas:

- Página web pública: Donde el público general encuentra información sobre el evento.
- Intranet: Permite a cada colegio dar de alta y modificar a sus participantes y equipos, y a la organización añadir, editar y eliminar toda la información del sistema.
- Aplicaciones móvil: iOS y Android. Permitirán acceder a los resultados y clasificaciones desde dispositivos móvil a lo largo del evento.

1.5 Estado del arte

1.5.1 - Aplicaciones existentes

Vistos los requerimientos de la Organización se buscan soluciones que satisfagan todas las necesidades.

Primero se analiza el Proyecto de Final de Carrera de VERÓNICA MIÑARRO PLAZA, del año 2014: APLICACIÓN WEB PARA LA GESTIÓN DE UN EVENTO DEPORTIVO [20] . En este proyecto se intenta resolver una situación parecida. Se requiere de un sistema para poder gestionar los participantes de una carrera, en el que cada participante se registra y pague una cuota. Además, ofrecer al público los tiempos y clasificaciones de los corredores a tiempo real. Se trabaja desde la planificación del proyecto desde cero, el análisis de requisitos, las especificaciones del sistema, la implementación en php, y la pasarela de pago a través de Papal. Pese a que la gestión de usuarios es parecida, los requisitos son tan diferentes que no se puede aprovechar más que el análisis de herramientas de desarrollo, los costes económicos y temporales.

También se analiza la web de gestión de licencias de la Federación Catalana de Básquet [19], creada por playoffinformatica. Al ser un software creado por una empresa profesional, cumple con todas las necesidades, es 100% personalizable y tendría todas las funcionalidades que requiere la Organización, pero el coste es excesivo. Dada la flexibilidad que exige la Organización, y el presupuesto marcado, la mejor opción es desarrollar un software a medida, del que se disponga el código fuente y conocimientos para modificarlo a placer.

1.5.2 - Tecnologías existentes

En lo que a tecnologías utilizadas, se explican más adelante en este documento, se decidió utilizar software de Microsoft como Visual Studio [3], SQL Server [4], ASP.Net [2], etc. por facilidad de integración con la infraestructura existente, pero también se estudiaron otras opciones.

Una opción descartada fue utilizar JSP de Java, pero debido a la falta de integración nativa en el Windows Server del que dispone la Organización se desestimó. Otra opción era utilizar tecnologías de client side scripting, como AngularJS, pero fue descartado tras unas pruebas por la complejidad en la configuración de los servicios web del servidor para ser utilizados por este sistema.

1.6 Legalidad vigente

La principal ley a tener en cuenta en este proyecto es la Ley Orgánica de Protección de Datos Personales[22]. Teniendo en cuenta que en la base de datos se almacenan datos personales (dni, nombres, datos de contacto, foto) de menores de edad se debe obligar a los colegios participantes a hacer firmar una autorización para gestionar y almacenar estos datos en el sistema. Asimismo, solo 2 personas de la organización han tenido acceso suficiente como para ver los datos personales, el responsable de competición, y el de acogidas. Cabe añadir que ningún dato personal relacionado con los equipos o los participantes sale de las fronteras españolas, ya que el servidor se encuentra en Santiago de Compostela. En caso de que los datos se almacenaran fuera de la Unión Europea deberíamos avisar a los usuarios.

Además, hay que tener en cuenta la directiva europea 2000/31/CE [24], ratificada por la Jefatura del Estado con la Ley 34/2002 [23], sobre el uso de cookies en internet. En nuestro proyecto usamos cookies para gestionar la sesión de usuario, para almacenar información sobre el idioma del usuario, y para los códigos de analítica web (Google Analytics).

2. Planificación inicial

2.1 Método de trabajo

Dada la cercanía con el cliente y la gran cantidad de trabajo a realizar en tan poco tiempo, utilizamos una "metodología ágil" para desarrollar el proyecto. Esta metodología permite desarrollar y publicar antes las funcionalidades que el cliente prioriza, aquellas que tienen más valor para él dado el calendario.

Desde el inicio del proyecto hay una reunión semanal con la Organización para valorar el avance del desarrollo y controlar que se cumplan los plazos. Las actas de estas reuniones, a veces innecesarias, sirven para definir el trabajo a corto plazo e ir estimando el alcance real del proyecto. Estas reuniones forman parte de las reuniones ordinarias de la Comisión Olímpica.

En este marco, la metodología seguida es "Extreme Programing" [1], en adelante XP. Ésta metodología se basa en una relación constante entre cliente y equipo de desarrollo, mucha flexibilidad ante el cambio, tanto de funcionalidades como de prioridades, y en la realización de pruebas unitarias en iteraciones muy pequeñas, para que el cliente pueda participar del crecimiento del proyecto.

2.2 Fases

El proyecto se divide en 4 fases: promoción, gestión, validación de datos y competición. Debido a que el único recurso humano disponible es el autor del proyecto, las tareas se ejecutan secuencialmente, teniendo en cuenta las posibles dependencias y las especificaciones marcadas por la Organización. Cada tarea se valida individualmente, y cada fase se valida con el cliente, dejando un periodo para pruebas y modificaciones antes de presentar cara al público las nuevas funcionalidades.

Para ver el desglose completo de las tareas, se adjunta el ANEXO I: Planificación temporal.

Fase 1 - Promoción

Se trabaja solo la web pública, desde el 1 de enero de 2015, hasta el 31 de marzo de 2015. Las tareas ejecutadas son aquellas relacionadas con la propaganda y la captación de voluntarios y familias de acogida.

Fase 2 - Gestión

Se trabaja sobre todo en la web privada, para dar a los colegios la herramienta con la que gestionar sus equipos y participantes, desde el 1 de abril de 2015, hasta el 10 de septiembre de 2015.

Fase 3 – Validación de datos

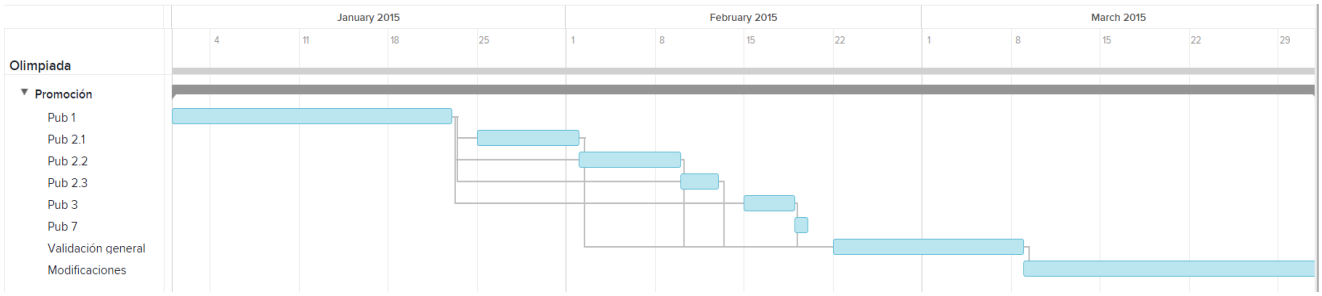
Se trabaja en ambas webs. En la web privada, para dar a la Organización la herramienta con la que gestionar visualizar, y validar la información introducida por los colegios. En la web pública, se añade información de interés general. Desde el 14 de septiembre de 2015, hasta el 31 de diciembre de 2015.

Fase 4 – Competición

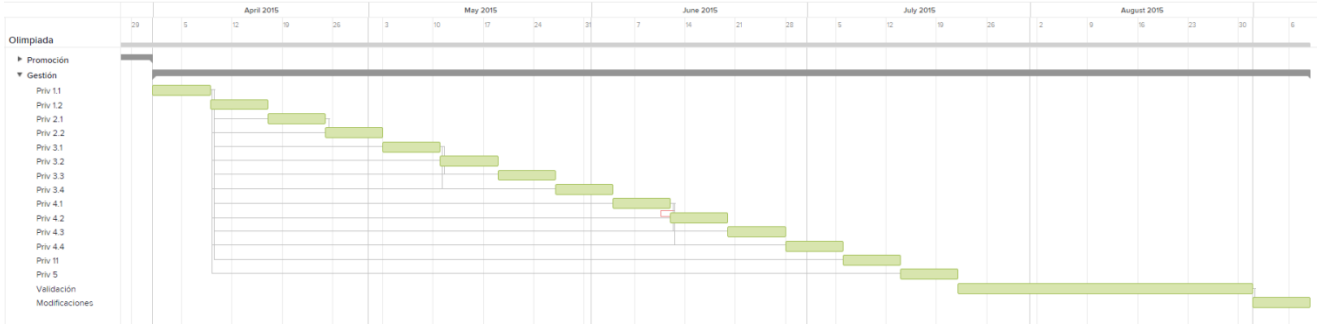
Se trabaja en ambas webs y en las aplicaciones móviles. En la web privada, para dar a la Organización la herramienta con la que gestionar los partidos. En la web pública, se añade información relativa a la competición. Se crean las aplicaciones móviles. Desde el 1 de enero de 2016, hasta el 24 de enero de 2016.

2.3 Diagrama de Gantt

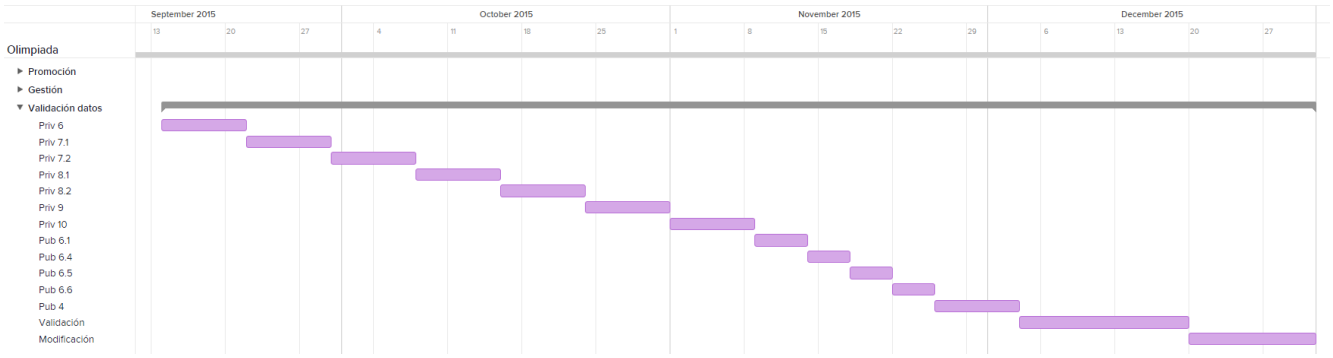
Fase 1 – Promoción



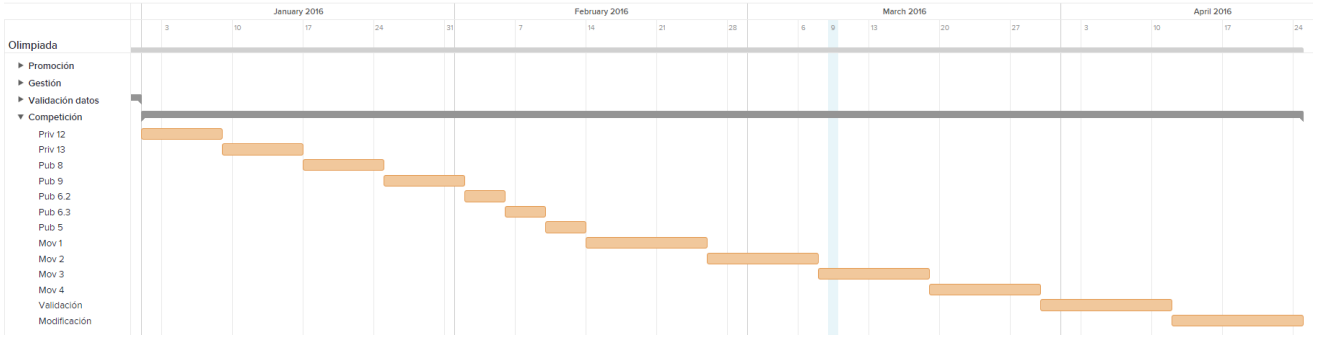
Fase 2 – Gestión



Fase 3 – Validación de datos



Fase 4 – Competición



2.4 Alternativas previstas

Ante la alta probabilidad de retrasos, los desarrollos que menos interesan a la Organización son las aplicaciones móviles, ya que sus contenidos son consultables desde la web, y los tiempos de ejecución asignados solo permitirían aplicaciones muy simples y poco elaboradas. En previsión a esta posible desviación, desde el inicio se usa una plantilla "responsive", es decir, que se adapta a dispositivos móviles. Este plan de acción permite ahorrar un mes y medio de desarrollo, que puede ser dedicado a tareas que hayan quedado rezagadas.

Para el resto de imprevistos menores, las horas de dedicación a "Validación general de fase" y "Modificación y publicación" son flexibles. Así conseguimos acabar las fases a tiempo y cumplir con un cierto margen de seguridad con los siguientes plazos impuestos por la organización:

- Junio 2015 – Información básica en la web pública (estimado para marzo 2015).
- Enero 2016 – Gestión de equipos y participantes (estimado para noviembre 2015).
- Marzo 2016 – Listas y validación de equipos y participantes (estimado para diciembre 2015).
- Abril 2016 – Gestión de competición y partidos (estimado para abril 2016).
- Mayo 2016 – Todo funcionando (estimado para abril 2016).

2.5 Planificación económica

Para este documento, vamos a partir siguiente supuesto: El cálculo de costes del proyecto se desarrolla completamente al margen de la Universidad Politécnica de Cataluña, eliminando las ventajas y descuentos de los que disfrutamos los miembros de la comunidad educativa. Con esto, intento que el resultado sea útil fuera del mundo académico. En la conclusión se especifica, además, el coste real, a modo de comparación. En ambos casos, y por coherencia con la planificación temporal, solo se cuenta con 1 trabajador, que ejerce los diferentes roles asociados al desarrollo del proyecto.

2.5.1 Gastos directos

Para estimar el coste en horas, y relacionarlo con el diagrama de Gantt de la planificación temporal, establecemos la fecha de inicio del proyecto el día 1 de enero de 2015, y de finalización el 8 de mayo de 2016. La duración del proyecto es de 70 semanas. Se establece una dedicación de media jornada, 10 horas a la semana, para tareas de diseño, especificación y testeo, con un coste de 20€/hora; 7 horas a la semana de implementación; más 3 horas de reunión semanal con la Organización de seguimiento y especificación de las tareas más prioritarias en el backlog, con un coste de 30 €/hora, ya que se imputan al jefe de proyecto. En cada una de las tareas participan los tres roles en estas proporciones.

Rol	Horas/semana	€/hora	€/semana	Semanas	TOTAL
Analista/diseñador	10	20	200 €	70	14.000 €
Programador	7	15	105 €	70	7.350 €
Jefe de proyecto	3	30	90 €	70	6.300 €
Total	20		**395 €**		**27.650 €**

Desglose de coste por fases:

Fase	Semanas	Coste
1 – Promoción	12.5	4.938€
2 – Gestión	23	9.085€
3 – Validación	16	6.320€
4 – Competición	18.5	7.308€
TOTAL	70	27.650€

En el ANEXO II: “Gestión Económica” se puede encontrar el desglose de costes por cada tarea.

2.5.2 Gastos indirectos

Se alquila un despacho cerca de la sede de la Organización, y se hace frente a diferentes facturas y gastos durante los 17 meses que dura el proyecto.

Concepto	Precio mensual	Precio total
Alquiler despacho	200€	3.400 €
Energía	40€	680 €
Agua	10€	170 €
Internet (Movistar Fibra 300)	34€	578 €
DinaHosting Windows	15€	255 €
DinaHosting Ampliación SQL Server	24€	408 €
Portátil Windows		650 €
Licencia Visual Studio 2013		646 €
Mac mini		550 €
Apple developer license		150 €
TOTAL		**7.487 €**

2.5.3 Imprevistos y contingencia

Como en cualquier proyecto, la planificación suele sufrir ligeras modificaciones durante su ejecución. La más típica es tener que hacer horas extra debido a cambios propuestos a posteriori, una mala dimensión de los recursos (bases de datos que se quedan pequeñas…) y equipos estropeados. Para poder cubrir cualquier situación inesperada, se pacta una partida de imprevistos del 10% de los gastos directos e indirectos, es decir, de 3.500€

Además, como plan de contingencia, en vez de aumentar el gasto se decidió reducir el alcance del proyecto, eliminando el Work Package 3 – Aplicaciones Móviles, que representa casi un 10% del tiempo de ejecución del proyecto, reduciendo además la partida de gastos indirectos 700 € derivados de la compra del Mac mini y la licencia de desarrollador de Apple, quedando así en 6.787 €

2.5.4 Total gastos

Con las cifras calculadas, se elabora el presupuesto estimado. Añado los costes reales que ha tenido el proyecto real, al desarrollarse como TFG, que solo la ampliación de la Base de datos, al contar equipamientos propios, licencias académicas o usar recursos de la Organización.

Partida	Presupuesto	Coste real ejecutado
Gastos directos	27.650 €	0 €
Gastos indirectos	7.487 €	198 €
Imprevistos	3.500 €	0 €
Contingencia	0 €	0 €
Total	**38.637 €**	198 €

3. Análisis de requisitos

3.1 Requisitos funcionales

3.1.1 Página pública

La página pública debe cumplir con los siguientes requisitos:

- El contenido debe estar accesible en castellano, catalán y euskera, a decisión del usuario. El sistema debe recordar esta preferencia.
- Debe existir un formulario de contacto accesible al público, para contactar con la Organización.
- Los enlaces a las redes sociales deben estar siempre visibles.
- El menú de navegación debe estar siempre visible.
- Los horarios, resultados y clasificaciones deben siempre actualizados.
- El formulario de solicitud de acogida debe contener todos los campos necesarios.
- La web debe tener un contenido *responsive*, es decir, amigable si se accede desde dispositivos móviles.

3.1.2 Página privada

La página privada debe cumplir con los siguientes requisitos:

- Se pedirá un usuario y contraseña para acceder.
- Todos los datos privados deben ser accesibles únicamente por la organización y el responsable de esos datos, incluidas todas las fotografías en las que aparezca un menor.

Si accede el responsable de un colegio:

- Los colegios disponen de un resumen de su información más relevante en la pantalla de inicio.
- Desde la pantalla de inicio puedo crear un equipo, o un participante.
- Los formularios de creación y edición de un participante son iguales.
- Al acceder a un equipo, se muestra la lista de participantes.

Si accede un responsable de la organización:

- La organización tiene acceso de lectura y escritura sobre todos los equipos y sus participantes.
- En la pantalla de inicio se muestra un resumen de los coordinadores y entrenadores inscritos.
- Se tiene acceso a la lista de solicitudes de acogida realizadas en el sistema.
- Se muestra una vista con todos los participantes, ordenados por equipos y colegios.

Si accede un voluntario de la organización:

- En la pantalla inicial se muestra un selector de deporte.
- Al acceder a un deporte se muestran todos los partidos disponibles para solicitar.
- Al solicitar un partido, el voluntario puede modificar su marcador, que será visible desde la página web pública en directo.

3.2 Requisitos no funcionales

Los requisitos no funcionales que debe cumplir el sistema son:

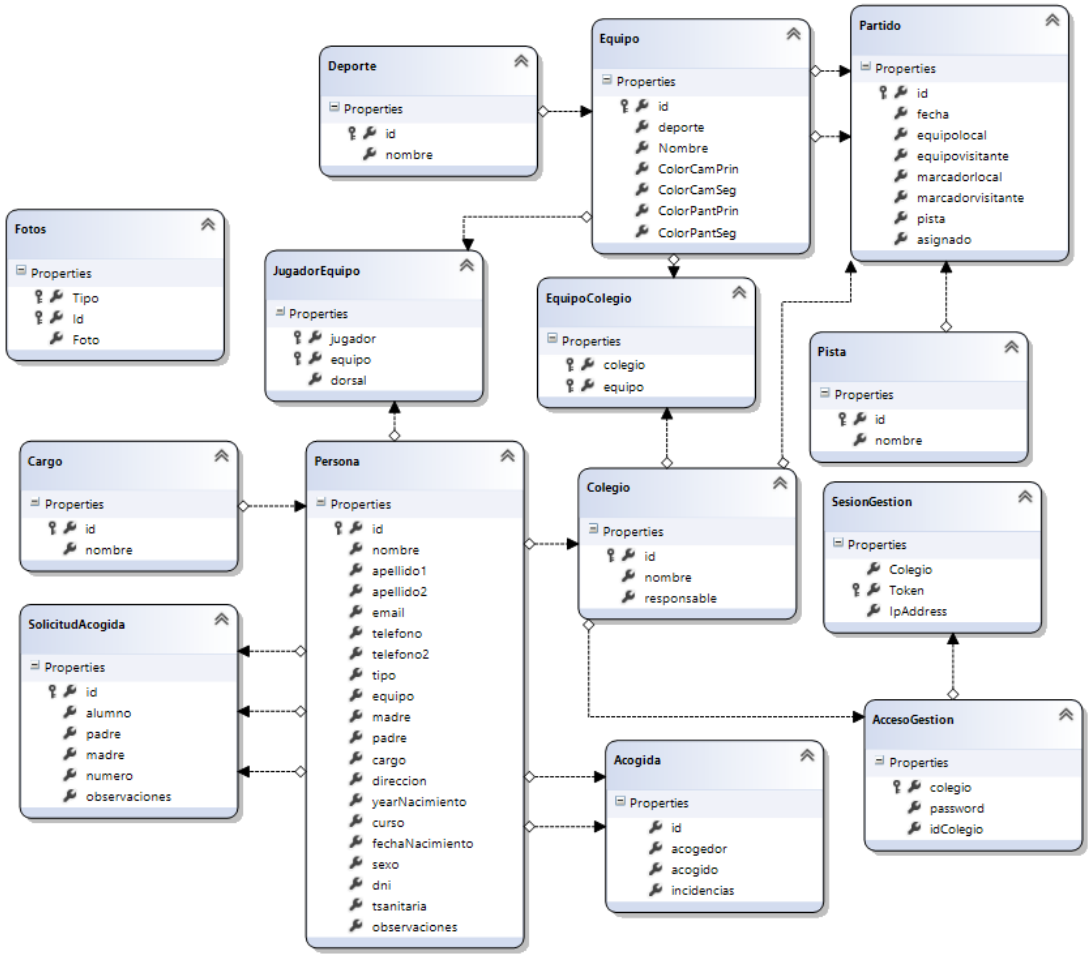
- **Multiplataforma**: La aplicación web debe poder ejecutarse y visualizarse correctamente en cualquier dispositivo, independientemente de la resolución de su pantalla, y en los principales navegadores: Chrome, Safari, Firefox e Internet Explorer.
- **Extensibilidad**: La aplicación web debe ser fácilmente ampliable, de manera que resulte sencillo añadir nuevas funcionalidades o modificar las actuales.
- **Usabilidad**: La aplicación debe ser intuitiva y fácil de utilizar, incluso para una persona con pocos conocimientos de informática.
- **Accesibilidad**: Las funcionalidades deben ser fáciles de localizar, y se debe minimizar el número de pasos para acceder a una funcionalidad determinada.
- **Ubicuidad**: Los datos deben ser accesibles desde cualquier dispositivo.

4. Especificación

4.1 Modelo conceptual

En este apartado mostraré qué tablas forman el sistema de información y cómo están relacionadas entre ellas mediante el diagrama de tablas normalizado. A continuación explicaré resumidamente qué representa cada una de estas tablas principales.

4.1.1 Tablas



- **Persona**: Es la tabla que almacena toda la información sobre personas, ya será en las solicitudes de acogida, en los participantes, entrenadores, responsables…
- **Colegio**: Almacena la información relativa a un colegio y sus equipos.
- **Equipo**: Información relativa a un equipo, sus participantes y equipos.
- **AccesoGestión**: Contiene las cuentas de acceso a la web privada.
- **SesiónGesitón**: Contiene las sesiones de acceso activas.
- **Partido**: Almacena la información sobre los partidos.

4.2 Actores

Se han definido los siguientes actores en el sistema:

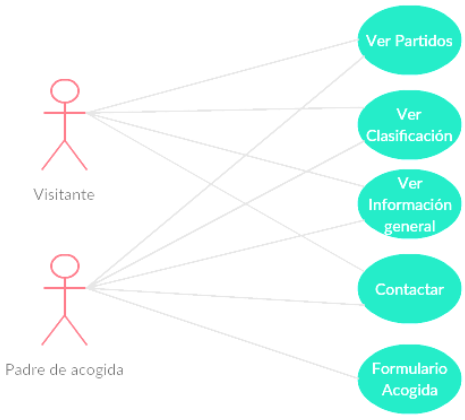
- **Visitante**: Es el usuario que accede a la web pública en busca de información. No tiene acceso a la web privada.
- **Padre de acogida**: Es el usuario que quiere rellenar el formulario de acogida. No tiene acceso a la web privada.
- **Voluntario**: Es el usuario que colabora con la organización durante el evento. Su tarea es mantener los marcadores actualizados. Tiene acceso al panel de partidos en la web privada. Se crean 9 cuentas de este tipo.
- **Responsable de colegio:** Es el usuario que cada colegio designa para introducir los datos de equipos y participantes. Existe una cuenta por colegio.
- **Administrador:** Tiene acceso a toda la información sobre equipos, participantes, solicitudes de acogida presentadas, etc. Existe una cuenta en total.

4.3 Casos de uso

En este apartado presentaré los diagramas de casos de uso. Para facilitar su comprensión, crearé un diagrama de casos de uso para cada área accesible desde la aplicación web: un diagrama para la web pública y otro para la web privada.

4.3.1 Web pública

Este es el diagrama de casos de uso de la web pública. Un visitante puede: Ver los partidos, ver la clasificación, ver la información general (Actividades, hoteles,…), acceder al formulario de contacto, etc, pero al rellenar el formulario de acogida, se entiende que ese usuario es un 'Padre de acogida'.



4.3.2 Web privada

Este es el diagrama de casos de uso de la web privada. El Log In es común, pero en función del cargo, se permite ver y realizar unas acciones concretas.

El responsable de un colegio solo tiene acceso a sus equipos y sus participantes. Puede dar de alta, modificar y eliminar equipos, asignarles participantes existentes, crear nuevos participantes o eliminarlos. Además puede designar un responsable de contacto del AMPA.

El administrador tiene capacidad para ver y editar los equipos y participantes de todos los colegios, además de listar equipos y participantes por colegios.

Los voluntarios pueden reservar un partido y editar su marcador.

5. Diseño

5.1 Tecnologías utilizadas

Para hablar de las tecnologías a utilizar, empezaremos desde los clientes, e iremos adentrándonos hasta la base de datos. Se han utilizado muchas tecnologías y herramientas de Microsoft debido a la facilidad de uso, integración entre ellas, la disponibilidad de un servidor web con Windows Server y conocimientos previos. En el punto anterior se explican alternativas que fueron descartadas.

5.1.1 Web

- HTML: HyperText Markup Language. Lenguaje estándar de definición de contenido. [5]
- CSS: Cascading Style Sheets. Lenguaje de definición de estilos para web. [6]
- Javascript & JQuery: Lenguajes de programación web de ejecución en el navegador del cliente.[7][8]

5.1.2 Comunicación cliente - servidor

- AJAX: Asynchronous JavaScript And XML. Técnica basada en javascript para la realización de llamadas HTTP. [9]
- HTTP: Hypertext Transfer Protocol. Protocolo de comunicación que permite la transferencia de información a través de internet. [21]
- JSON: JavaScript Object Notation. Formato de texto muy frecuente en la comunicación a través de servicios web. [10]

5.1.3 Servidor

- ASP.NET: Active Server Page. Framework de Microsoft para la creación dinámica de páginas en el lado servidor. [2]
- ASMX: Active Server Method. Mecanismo de ASP.NET que contiene un servicio web.[11]
- ASHX: Active Server Handler. Mecanismo de ASP.NET para controlar llamadas HTTP. En el proyecto se usa para la transmisión de las imágenes mediante llamadas HTTP, ya que no están guardadas en el sistema de ficheros, sino en una base de datos.
- IIS: Internet Information Service. Servicio de Windows que permite la publicación de páginas web.[12]
- C#: Lenguaje de programación orientado a la ejecución de aplicaciones. [14]

5.1.4 Comunicación servidor – base de datos

- LinQ: Language Integrated Queries. ORM que conecta una base de datos relacional con objetos C#.[13]

5.1.5 Base de datos

- SQL Server: Sistema de gestión de bases de datos relacionales. [4]
- SQL: Lenguaje de acceso y operación a bases de datos.

5.2 Herramientas de trabajo

Estas son las principales herramientas a utilizar durante el proyecto:

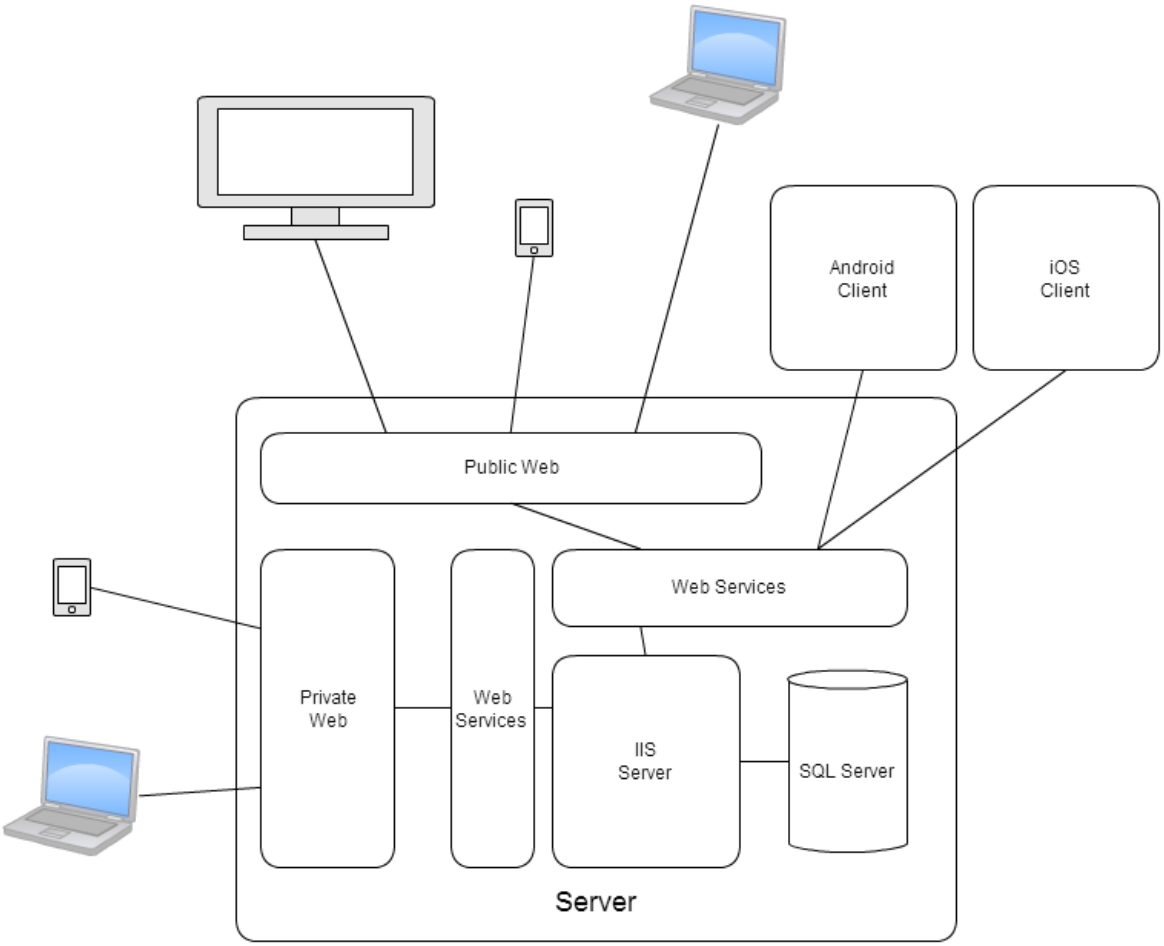
- Visual Studio 2013: IDE para la creación de las páginas web, y las aplicaciones de servidor. [3]
- SQL Server 2012: Motor de bases de datos. [4]
- SQL Management Studio: Herramienta para la gestión de bases de datos.
- Google Chrome: Navegador web con debugger.
- DinaServer: Herramienta de gestión de dominios y hostings de Dinahosting SL.
- Google Analytics: Herramienta de analítica web.
- MailChimp: Herramienta de envío masivo de correo.
- TeamGantt: Herramienta de planificación temporal de tareas.
- TortoiseSVN: Software cliente de control de versiones.

5.3 Arquitectura

El sistema se basa en 2 páginas web que se comunicarán con el servidor a través de servicios web. En el mismo servidor se alojarán las páginas, los servicios, el software de gestión y la base de datos. El esquema es el siguiente:

5.3.1 Web

Las páginas web están implementadas en HTML. En las páginas de contenido dinámico, inicialmente el cliente solo recibe la estructura de los datos a mostrar, y en cuanto se ha cargado la página, se realiza una llamada AJAX [9] para solicitar al servidor los contenidos. Por ejemplo, en la página de la clasificación, el navegador realiza la petición de la página y recibe, en html, un fichero con la página,



pero sin el contenido de los datos. A continuación, el código javascript realiza la llamada al servicio web para conseguir los datos de la clasificación.

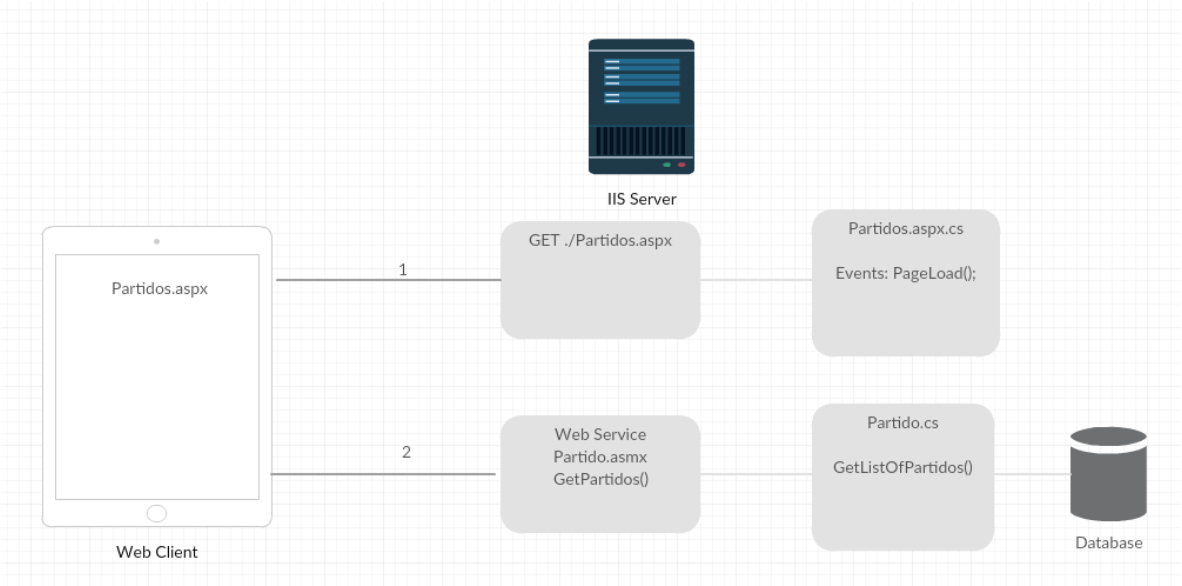
Con este sistema, se agiliza la carga de la página, ya que el usuario el usuario recibe enseguida el HTML de la página, y mientras espera a que lleguen los datos se puede mostrar un mensaje de carga. Además, la llamada AJAX al servicio web es pública, así que se permite que los alumnos de secundaria implementen sus propias aplicaciones y páginas con los datos reales para la asignatura de informática.

5.3.2 Servidor

El código de servidor está escrito en C#. Dividiremos entre la parte que genera y sirve los archivos html; y la parte que se encarga de los servicios web.

La parte que genera los ficheros html que se envían en las peticiones HTTP son los archivos ASPX. Estos archivos constan de 2 partes, la parte cliente, en la que se ve el código html que se va a transmitir, y la parte de servidor, con las funciones que se ejecutan para añadir contenido dinámico al html.

La parte que controla los servicios web son los archivos ASMX. Estos archivos contienen las funciones públicas que se exponen cara al público. Estas funciones públicas utilizan las clases privadas del dominio, que contienen la lógica de nuestra aplicación, y a su vez se comunican con la base de datos mediante LINQ, el componente de .NET diseñado para la consulta de datos de manera nativa.



5.4 Interfaz

Para hablar de la interfaz, volveremos a separar entre los dos sistemas, la web pública y la web privada. Este diseño ha pasado por varias etapas desde que se empezó a definir, iniciándose a partir de bocetos esbozados en papel, y pasando por diferentes versiones hasta llegar a la versión final. En este documento no se pretenden mostrar todas y cada una de las pantallas, sino algunos ejemplos para dar una idea de cómo han sido estructuradas y con qué propósitos.

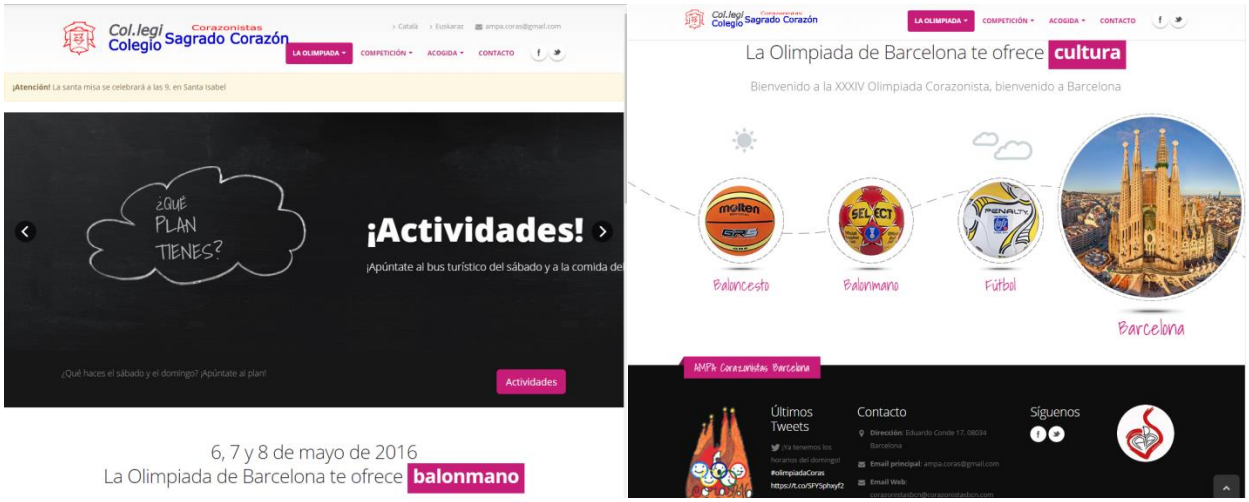
El diseño de la interfaz de usuario se ha realizado teniendo en cuenta los siguientes requisitos no funcionales:

- **Usabilidad**: La aplicación debe ser intuitiva y fácil de utilizar, incluso para una persona con pocos conocimientos de informática.
- **Accesibilidad**: Las funcionalidades deben ser fáciles de localizar, y se debe minimizar el número de pasos para acceder a una funcionalidad determinada.
- **Interfaz atractiva**: La interfaz debe ser atractiva, puesto que a través de la aplicación pretendemos vender un producto.
- **Multiplataforma**: La aplicación web debe poder ejecutarse y visualizarse correctamente en cualquier dispositivo, independientemente de la resolución de su pantalla, y en los principales navegadores: Chrome, Firefox e Internet Explorer.

5.4.1 Web pública

La web pública debe servir como referente a la hora de encontrar la información relevante del evento de manera ágil, así que se ha decidido incluir los datos más importantes, como fechas y avisos, en la pantalla principal. Además, a través del menú siempre visible en la parte superior podemos acceder a todas las secciones. La web está disponible en la siguiente dirección: olimpiadas.corazonistasbcn.com

En la pantalla inicial podemos ver:



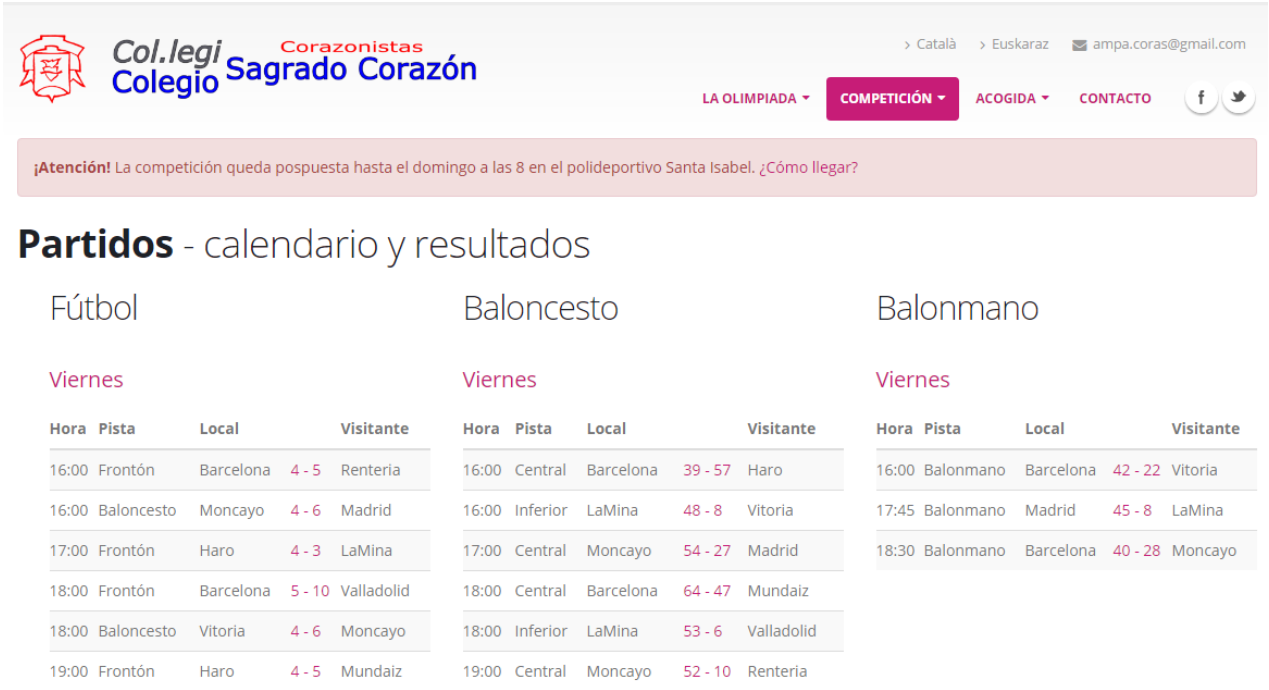
En la parte superior podemos ver siempre el menú de navegación, y la sección en la que estamos se muestra activa. Podemos ver arriba a la derecha un selector de idioma. Al seleccionar un idioma, todos los textos se mostrarán traducidos, y el sistema recordará esta preferencia durante toda la navegación.

Lo siguiente que vemos es, en amarillo, la sección de avisos. Si el aviso es importante, como un cambio de horario, se muestra en rojo, mientras que si es una notificación menor, como un recordatorio, se muestra en amarillo.

Después, sobre fondo 'negro pizarra' se muestra la sección de noticias generales. Van pasando las noticias generales con ilustraciones o dibujos y enlaces de referencia. En este caso vemos la promoción de las actividades organizadas para las familias.

Se pueden ver más secciones de información general, hasta el contacto, con los tweets recientes y enlaces a las redes sociales.

A continuación mostraré un ejemplo de página dinámica, los resultados. Como se ha explicado anteriormente, primero se carga la página html y después los partidos y sus resultado.

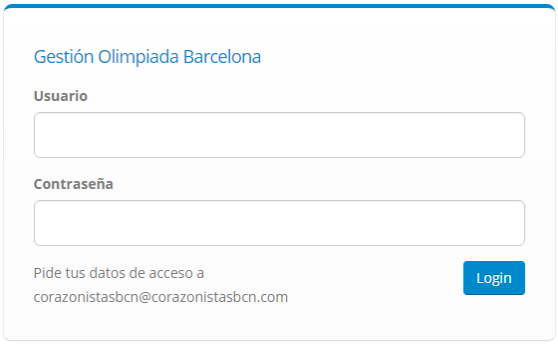


Podemos ver un aviso urgente en la parte superior, en rojo. Estos avisos sirven para mantener informados a los usuarios de cualquier cambio en la planificación.

5.4.2 Web privada

La web privada solo es accesible con usuario y contraseña, y se muestra un contenido diferente según el acceso. La web está disponible en la siguiente dirección: core.olimpiadas.corazonistasbcn.com.

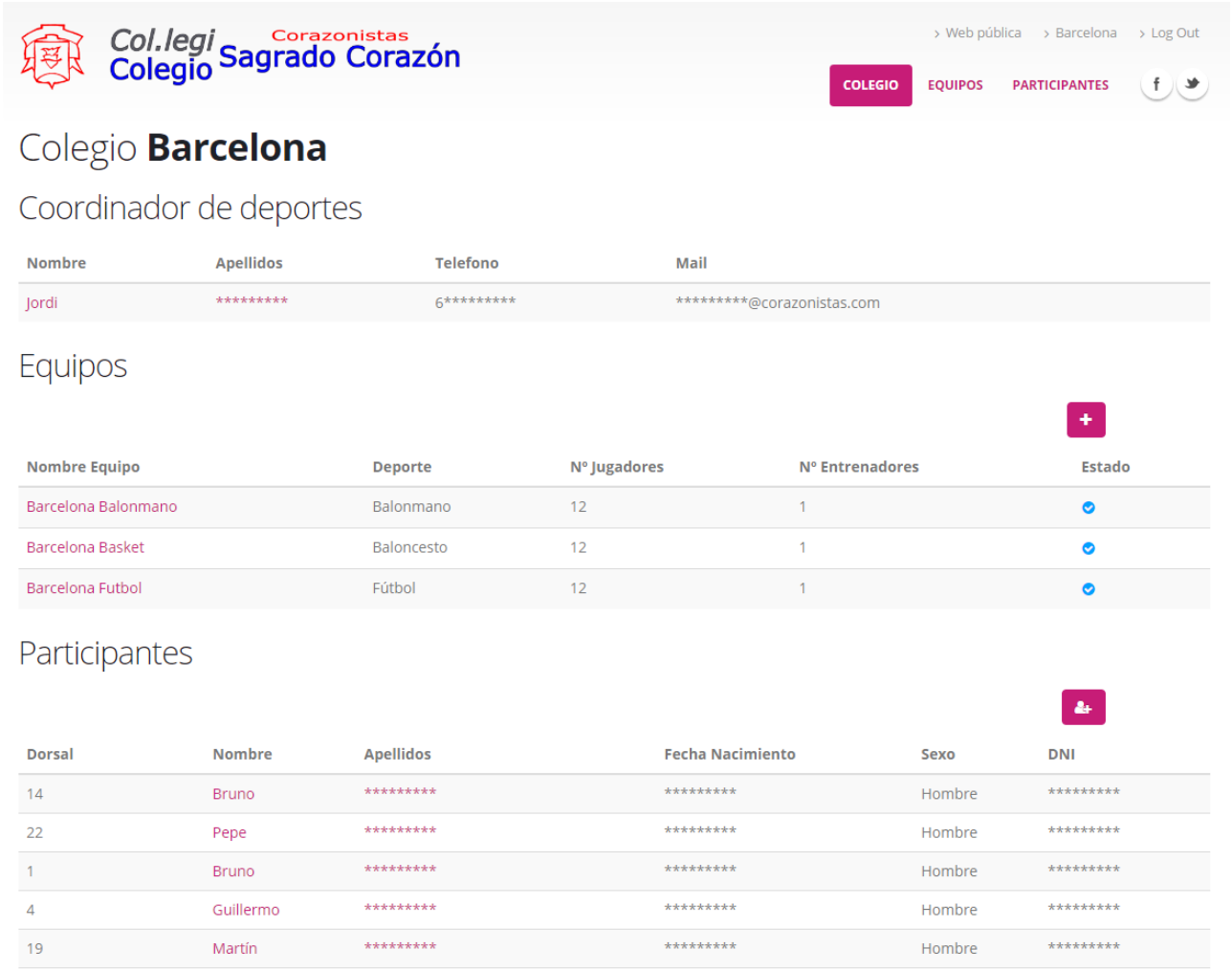
Lo primero que se muestra es la pantalla de LogIn. Al tener todas las cuentas creadas previamente, no existe un formulario de registro.



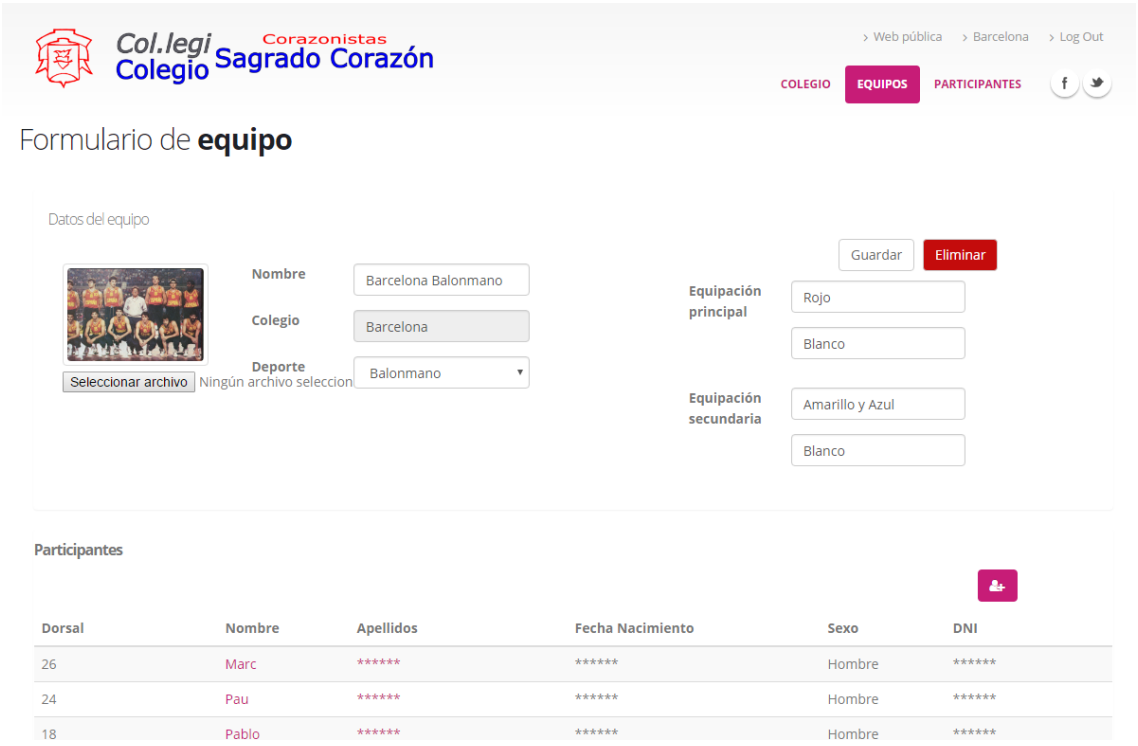
Hay 3 tipos de acceso distintos: el colegio, el voluntario, y la organización. Todas las cuentas han sido creadas previamente, y cada responsable ha recibido en mano un documento con las instrucciones de acceso y un video tutorial de uso, disponible aquí https://goo.gl/TNaCwV

Vista Colegio

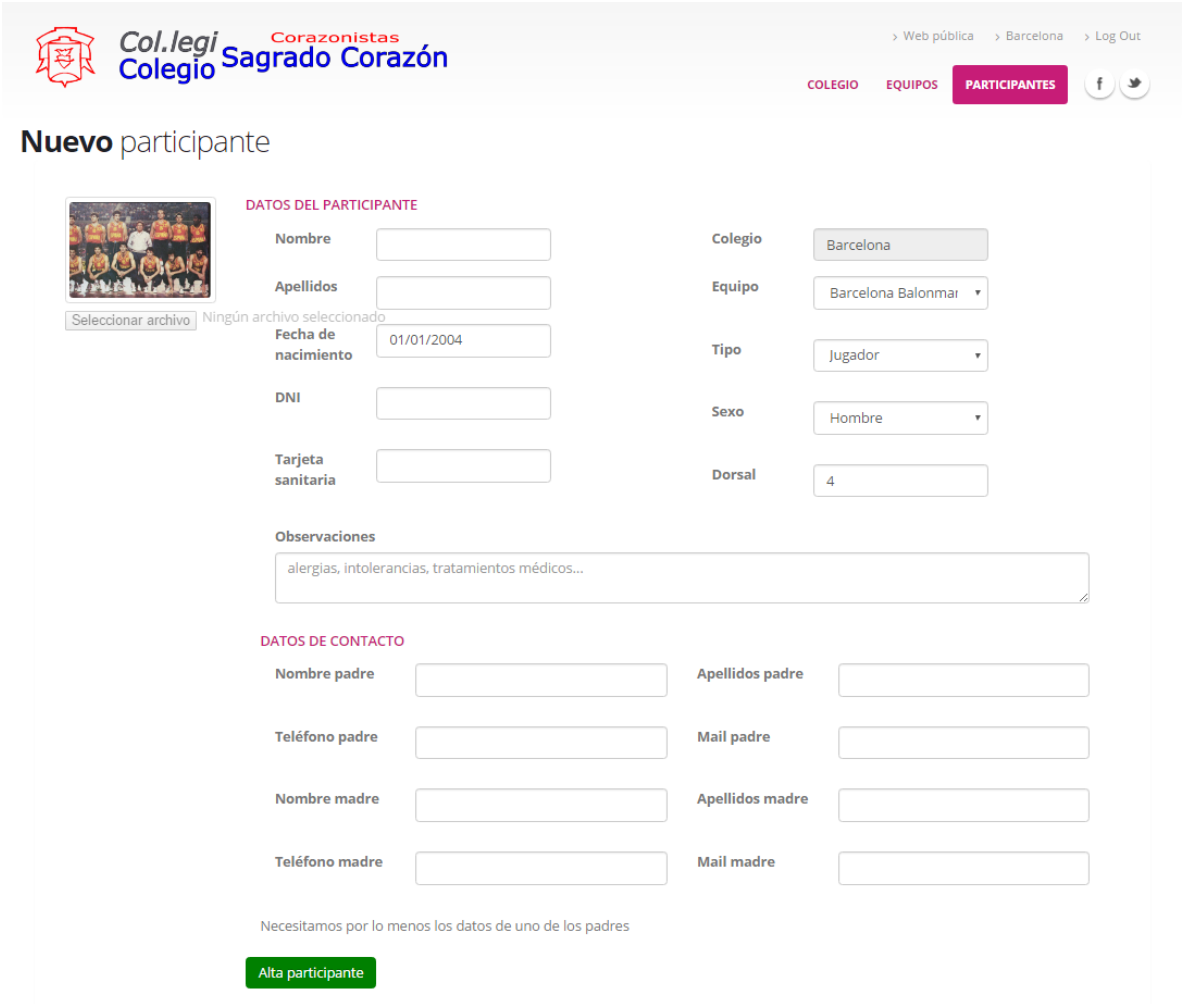
Primero accedemos a la vista de colegio, con el de Barcelona. En la pantalla inicial vemos un resumen, los datos del coordinador, los equipos dados de alta, y una lista de participantes. Por razones evidentes, se han censurado los datos de carácter personal de la imagen.



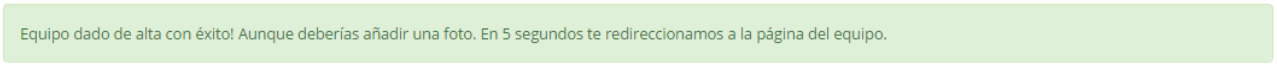
Con los botones + se pueden crear equipos nuevos y participantes nuevos, y los textos en fucsia son enlaces a la ficha individual. Si accedes a un equipo, te muestra los datos del equipo y sus participantes, y si accedes a un participante se muestran sus datos personales y los de contacto de los padres. A continuación se ve la pantalla de edición de un equipo:



El formulario de edición de un equipo o jugador, es igual que el de creación. A continuación, se muestra un formulario de alta de participante.



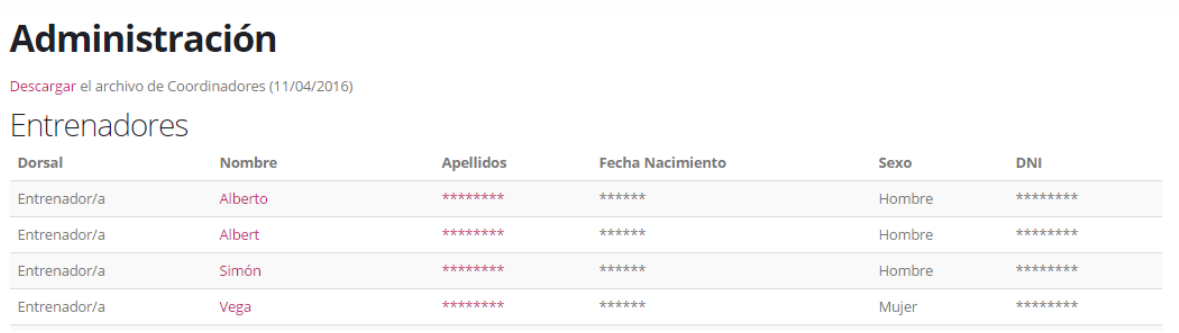
Para ayudar a los usuarios con los formularios, en la parte superior aparecen diferentes avisos con información sobre el estado del proceso a realizar. Por ejemplo, al dar de alta un nuevo equipo sin adjuntar una foto podemos ver:



Y si se produce un error al dar de alta un participante, porque falta algún dato importante:

Vista Administrador

Si se accede en modo Administrador, en la pantalla principal veremos, a petición de la Organización, un listado de los entrenadores dados de alta en el sistema:



Administración

Descargar el archivo de Coordinadores (11/04/2016)

Entrenadores

Dorsal	Nombre	Apellidos	Fecha Nacimiento	Sexo	DNI
Entrenador/a	Alberto	********	******	Hombre	********
Entrenador/a	Albert	********	******	Hombre	********
Entrenador/a	Simón	********	******	Hombre	********
Entrenador/a	Vega	********	******	Mujer	********

En el apartado “Solicitudes de acogida” se ve un listado de todas las solicitudes de acogida realizadas en la página pública, para que los responsables puedan adaptar las necesidades de los niños (alergias, amistades…) con las ofertas de cada familia (posibilidad de acoger a celíacos, solo chicos…).

Listado de **acogidas** recibidas

id	Alumno	Padre	Madre	Plazas	Solicitud
19	Raul****** 4EI	Raul****** ****** ******	Shireem****** ****** ******	1	Si puede ser que el niño tenga la misma edad que el de ******** .Gracias
21	Martina****** 3EI	Natxo****** ****** ******	Beatriz****** ****** ******	2	Es una familia que viene del país vasco.
23	Alvaro****** 6EP	Roberto****** ****** ******	Rocío****** ****** ******	2	
29	Adriana y Luis****** 4EP	Sergio****** ****** ******	Gloria****** ****** ******	1	Podemos acoger a un niño-a celíaco. Yo, Gloria, soy celíaca y en casa lo tenemos muy bien preparado para que no exista contaminación cruzada, casi todo lo hacemos libre de gluten.Gracias.Gloria.
30	Rosario ****** 6EP	****** ****** ******	susana ****** ****** ******	1	Por favor preferiría a coger a una niña que a un niño. Gracias
31	Alex****** 2ES	Daniel****** ****** ******	Silvia****** ****** ******	1	Nosotros quisiéramos acoger a una NIÑA.Al tener tres hijos varones y una niña, dormirían los tres chicos juntos y la niña de acogida en otra habitación con mi hija.Gracias

En el apartado validación de equipos se puede ver un resumen de equipos por colegio. Esta pantalla es muy útil para conocer el estado general, número de participantes totales, si algún equipo tiene más participantes de los que dijo inicialmente…

Validación de **equipos** y **participantes**

Ver listado con todos los detalles

Barcelona

Nombre Equipo	Deporte	Nº Jugadores	Nº Entrenadores	Estado
Barcelona Balonmano	Balonmano	12	1	
Barcelona Basket	Baloncesto	12	1	
Barcelona Futbol	Fútbol	12	1	

Haro

Nombre Equipo	Deporte	Nº Jugadores	Nº Entrenadores	Estado
Haro Baloncesto Femenino	Baloncesto	9	2	
Haro Fútbol Sala	Fútbol	8	2	

En “Ver listado con todos los detalles” veremos el mismo listado pero con la lista de participantes de cada equipo. Esta vista es muy útil para sacar listados en pdf.

Validación de **equipos** y **participantes**

Ver listado reducido

Madrid

Fútbol Madrid

Dorsal	Nombre	Apellidos	Fecha Nacimiento	Sexo	Observaciones
9	Javier	*****	*****	Hombre	Alergia al polen. Tratamiento: Ventolín a demanda; Pulmicort cuando tiene crisis (1 pulverización/8 horas); Rinialer (1 pastilla en desayuno).
11	Javier	*****	*****	Hombre	Se alojará en casa de sus tíos, exalumnos de Barcelona
15	Isaac	*****	*****	Hombre	
16	Alvaro	*****	*****	Hombre	Sin alergias
Entrenador/a	Jorge	*****	*****	Hombre	

Moncayo

Baloncesto Moncayo

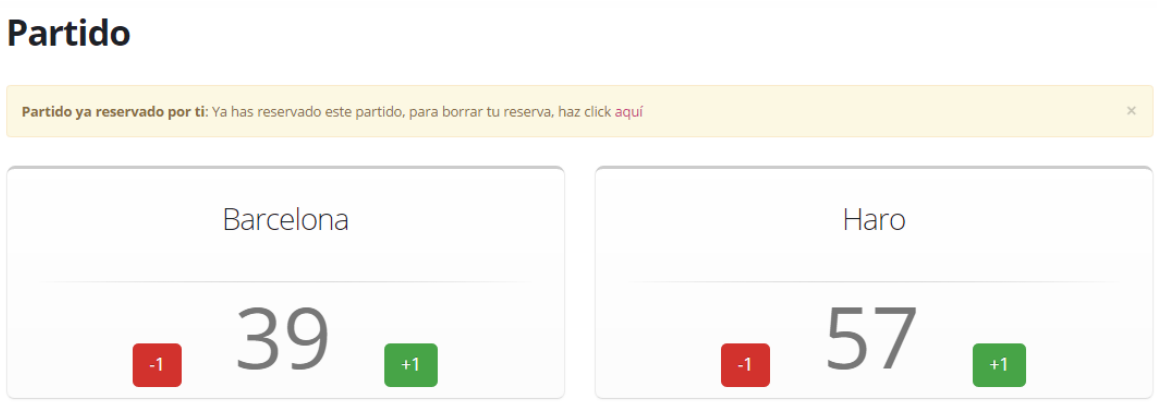
Dorsal	Nombre	Apellidos	Fecha Nacimiento	Sexo	Observaciones
17	OLGA	*****	*****	Mujer	
10	PAULA	*****	*****	Mujer	Alergia ocular al polen, si empieza el proceso hay que suministrarle un colirio en los ojos y en ocasiones un jarabe. A las olimpiadas llevará el tratamiento adecuado.
61	CARLOTA	*****	*****	Mujer	Alergia a las graminias, jarabe loratodile, aniestaminico en gotas opatanol para los ojos
64	PALOMA	*****	*****	Mujer	NO PUEDE TOMAR LECHE

Vista Voluntario

El modo ‘voluntario’ está para que un grupo de alumnos de 4º de Eso que colaboran con la organización mantengan actualizados los marcadores durante el evento. Estos voluntarios disponen de unas tablets cedidas por el colegio, y de sus propios dispositivos móviles. Cada uno tiene una cuenta diferente, y al acceder a la plataforma, escogen un deporte y un partido. Al reservarlo, ningun otro voluntario puede acceder . Este voluntario podrá ver las siguientes pantallas:

Con esta pantalla escogen el deporte. Normalmente, un voluntario conoce las normas de un deporte, ya sea porque es o ha sido jugador, o porque ha recibido formación específica. En cualquier caso, un voluntario puede cambiar de deporte cuando quiera.

Una vez seleccionado el deporte, se pueden ver los partidos disponibles. En este caso, el voluntario ya había reservado el partido de baloncesto de las 4, entre Barcelona y Haro, con lo que al acceder podrá ver:



El voluntario dispone de 4 botones con los que controla el marcador. Cada 5 segundos, y solo si ha habido algún cambio, se actualiza la base de datos con los resultados aportados por los voluntarios, con lo que en todo momento el público y la organización pueden disponer de esta información en tiempo real.

6. Implementación

En este apartado explicaré los aspectos más relevantes o destacables de la implementación del proyecto. Por lo tanto, aunque se han implementado todas las funcionalidades nombradas en esta documentación, en este apartado sólo se hará mención a una, en la que intervienen todas las capas, desde el código de cliente, los servicios web, las clases del dominio, la interacción con la base de datos y ésta misma. Vamos a basarnos en la funcionalidad: "Dar de alta a un participante en el equipo de un colegio".

6.1 Código cliente

Antes de empezar, hay que aclarar que ésta es una función que requiere haber hecho LogIn. Para mantenerse identificado en todo momento, al hacer LogIn el cliente recibe un token de sesión, un código alfanumérico que debe incluir en cada petición que requiera autentificación.

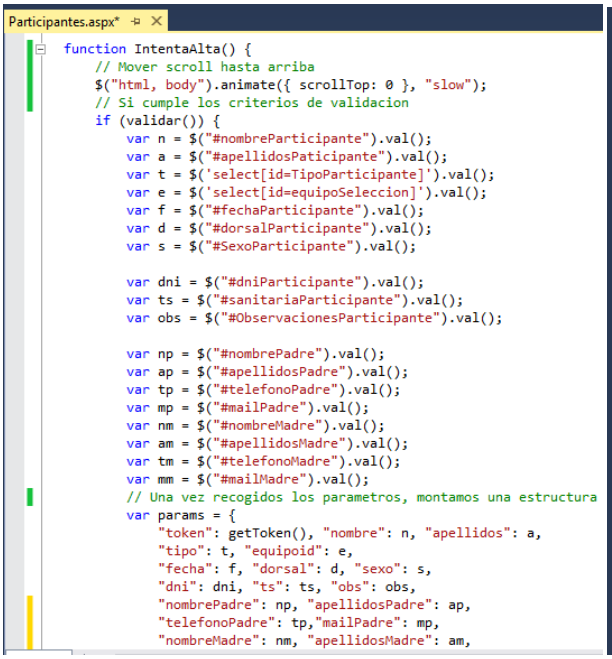
Veamos un fragmento de código HTML del formulario:

Participantes.aspx*

```
<div class="panel-body">
    <div class="col-md-2">
        <span class="img-thumbnail">
            <img id="FotoPlace" class="img-responsive" src="img/a.jpg" alt="Foto">
        </span>
        <form id="afoto" runat="server" >
            <input id="fotosube" type="file" accept=".jpg" style="outline: none; opacity: 0.5;" />
        </form>
    </div>
        <div class="col-md-10">
            <h5>Datos del participante</h5>
        <div class="col-md-6">

            <div class="form-group">
                <label class="col-md-3 control-label" for="inputDefault">Nombre</label>
                <div class="col-md-6">
                    <input type="text" class="form-control" id="nombreParticipante" value="">
                </div>
            </div>
            <div class="form-group">
                <label class="col-md-3 control-label" for="inputDefault">Apellidos</label>
                <div class="col-md-6">
                    <input type="text" class="form-control" id="apellidosPaticipante" value="">
                </div>
            </div>
            <div class="form-group">
                <label class="col-md-3 control-label" for="inputDefault">Fecha de nacimiento</label>
                <div class="col-md-6">
                    <input type="date" class="form-control mb-md" id="fechaParticipante" value="2004-01-01">
                </div>
            </div>
```

En este fragmento podemos ver parte del código correspondiente a los datos del participante. Cada input tiene un campo 'type' para que el navegador sepa qué tipo de datos esperar, un campo 'class' que le indica el estilo que se debe usar, un campo 'id' que sirve para identificar el input en el documento, y 'value' que es el valor asignado.

Una vez se ha mostrado la página, el usuario utiliza el botón “Alta participante” para enviar al servidor los datos. En este momento se ejecuta el siguiente código javascript.

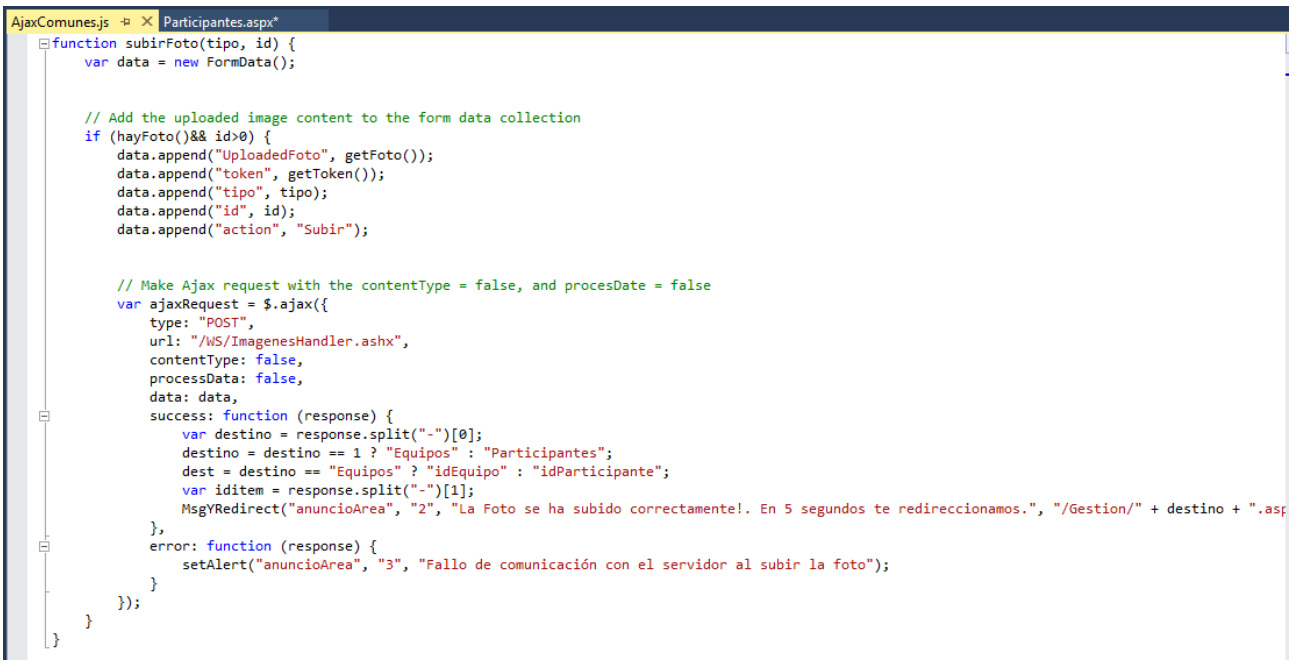


```
Participantes.aspx*
function IntentaAlta() {
    // Mover scroll hasta arriba
    $("html, body").animate({ scrollTop: 0 }, "slow");
    // Si cumple los criterios de validacion
    if (validar()) {
        var n = $("#nombreParticipante").val();
        var a = $("#apellidosPaticipante").val();
        var t = $('select[id=TipoParticipante]').val();
        var e = $('select[id=equipoSeleccion]').val();
        var f = $("#fechaParticipante").val();
        var d = $("#dorsalParticipante").val();
        var s = $("#SexoParticipante").val();

        var dni = $("#dniParticipante").val();
        var ts = $("#sanitariaParticipante").val();
        var obs = $("#ObservacionesParticipante").val();

        var np = $("#nombrePadre").val();
        var ap = $("#apellidosPadre").val();
        var tp = $("#telefonoPadre").val();
        var mp = $("#mailPadre").val();
        var nm = $("#nombreMadre").val();
        var am = $("#apellidosMadre").val();
        var tm = $("#telefonoMadre").val();
        var mm = $("#mailMadre").val();
        // Una vez recogidos los parametros, montamos una estructura
        var params = {
            "token": getToken(), "nombre": n, "apellidos": a,
            "tipo": t, "equipoid": e,
            "fecha": f, "dorsal": d, "sexo": s,
            "dni": dni, "ts": ts, "obs": obs,
            "nombrePadre": np, "apellidosPadre": ap,
            "telefonoPadre": tp,"mailPadre": mp,
            "nombreMadre": nm, "apellidosMadre": am,
```

```
Participantes.aspx*
// Informar al usuario de que se está enviando
setAlert("anuncioArea", "1", "Enviando información");
// Deshabilitamos el boton
$("#botonGuardar").prop('disabled', true);
// Llamada ajax al servidor
$.ajax({
    url: "/WS/Participante.asmx/NuevoParticipante",
    type: "POST",
    contentType: "application/json; charset=utf-8",
    dataType: "json",
    data: JSON.stringify(params),
    crossDomain: false,
    context: "",

    success: function (response) {
        var status = response.d;
        if (status > 0) {
            if (hayFoto()) {
                setAlert("anuncioArea", "2", "Participante creado con éx
                subirFoto("Participante", status);
            } else {
                MsgYRedirect("anuncioArea", "2", "Participante dado de a
            }
        }
        else setAlert("anuncioArea", "3", "Ups! Algo raro ha pasado.");

    },
    error: function (response) {

        setAlert("anuncioArea", "3", "Ups! Algo raro ha pasado.");

        $("#botonGuardar").prop('disabled', false);
    }
});
```

Lo primero que se hace es mover el ‘scroll’ hasta arriba. Luego se validan los campos, si hubiera algún error o campo obligatorio vacío, se mostraría un mensaje al usuario y no se seguiría. Si se supera la validación, Almacenamos todos los parámetros en la variable ‘params’. Mostramos un mensaje de aviso por pantalla y deshabilitamos el botón de enviar, y realizamos la llamada AJAX al servidor. Se definen los parámetros de la llamada HTTP, como la url (relativa desde la base de la aplicación), el tipo de llamada, POST, el tipo de contenido, los parámetros, y las acciones a seguir en caso de éxito o de fallo.

En caso de que la llamada sea un éxito, se muestra un mensaje en verde, y se empieza a subir la imagen del formulario. Las imágenes las tratamos a parte por su complejidad. Las imágenes no van a un servicio web ASMX sino a un HTTP Handler genérico ASHX. Se transfiere en un mensaje multipart que divide el contenido del payload (la imagen) en múltiples trocitos, que se recomponen en el otro extremo.



```
AjaxComunes.js | Participantes.aspx*
function subirFoto(tipo, id) {
    var data = new FormData();


    // Add the uploaded image content to the form data collection
    if (hayFoto()&& id>0) {
        data.append("UploadedFoto", getFoto());
        data.append("token", getToken());
        data.append("tipo", tipo);
        data.append("id", id);
        data.append("action", "Subir");


        // Make Ajax request with the contentType = false, and procesDate = false
        var ajaxRequest = $.ajax({
            type: "POST",
            url: "/WS/ImagenesHandler.ashx",
            contentType: false,
            processData: false,
            data: data,
            success: function (response) {
                var destino = response.split("-")[0];
                destino = destino == 1 ? "Equipos" : "Participantes";
                dest = destino == "Equipos" ? "idEquipo" : "idParticipante";
                var iditem = response.split("-")[1];
                MsgYRedirect("anuncioArea", "2", "La Foto se ha subido correctamente!. En 5 segundos te redireccionamos.", "/Gestion/" + destino + ".asp
            },
            error: function (response) {
                setAlert("anuncioArea", "3", "Fallo de comunicación con el servidor al subir la foto");
            }
        });
    }
}
```

6.2 Web service

Cuando el cliente realiza una llamada AJAX a un servicio web, tiene que definir los parámetros con el mismo nombre y del mismo tipo que aparezca en la definición de la función. Los servicios web del tipo SOAP, tienen que definir la lista de funciones públicas con sus parámetros en un fichero WSDL, para que sean accesibles. El framework .NET, utilizado en este proyecto, nos facilita la tarea, ya que cada vez que se compila el proyecto, se autogenera el fichero WSDL, de manera que no tenemos que ir actualizándolo a cada cambio. La implementación del servicio que hemos invocado en el fichero Participantes.asmx antes es la siguiente:

```
[WebMethod]
public string NuevoParticipante(string token, string nombre, string apellidos, int tipo, int equipoid, string fecha, int dorsal, string sexo, s
{
    try
    {
        //Context.Response.StatusCode = 418;
        //return "Error: Se ha cerrado el plazo. Contacta con la Organización" ;

        // A partir de la IP y del token obtenemos el Colegio que está realizando la petición
        string ipAddress = System.Web.HttpContext.Current.Request.UserHostAddress;
        int idCole = DomainModel.SesionGestion.getInfo(token, ipAddress);
        DomainModel.Colegio cole = new DomainModel.Colegio(idCole);
        DomainModel.Equipo equipo = cole.getEquipo(equipoid);
        // Conseguimos el equipo y le añadimos un participante
        DateTime f = Convert.ToDateTime(fecha);
        if (tipo == 3) dorsal = 7777;
        DomainModel.Participante p = equipo.addParticipante(nombre.Trim(), apellidos.Trim(), f, dorsal, sexo.Trim(), dni.Trim(), ts.Trim(), obs
        p.setPadreData(nombrePadre.Trim(), apellidosPadre.Trim(), telefonoPadre.Trim(), mailPadre.Trim());
        p.setMadreData(nombreMadre.Trim(), apellidosMadre.Trim(), telefonoMadre.Trim(), mailMadre.Trim());
        // Guardamos el participante en la base de datos y devolvemos su id
        p.Save();
        return p.getId().ToString();

    }
    catch (Exception ex)
    {
        Context.Response.StatusCode = 418;
        return "Error: " + ex.Message;
    }

}
```

Obtenemos la dirección IP de la petición http y el token de sesión. A partir de estos dos datos, sabemos si es un usuario válido y de qué colegio es. Cargamos el equipo referido en los parámetros, y le añadimos al nuevo jugador. Después guardamos en disco y finalmente devolvemos su id. En caso de recibir alguna excepción durante este proceso, devolveríamos un código de error acompañado de un mensaje.

Tal como hemos visto, la carga de imágenes es especial, ya que se realiza a través de un *handler.* Cabe añadir que la clase que gestiona los web services, hereda de esta.

```
namespace Olimpiadas.Core.Gestion
    public class ImagenesHandler : IHttpHandler
    {
        public void ProcessRequest(HttpContext context)
        {
            try
            {
                String token = context.Request["token"];
                int id = int.Parse(context.Request["id"]);
                String tipo = context.Request["tipo"];
                String action = context.Request["action"];

                if (action == "Subir")
                {
                    string ipAddress = System.Web.HttpContext.Current.Request.UserHostAddress;
                    int idCole = DomainModel.SesionGestion.getInfo(token, ipAddress);
                    DomainModel.Colegio cole = new DomainModel.Colegio(idCole);
                    int t = tipo == "Equipo" ? 1 : 2;
                    cole.addFoto(t, id, context.Request.Files.Get(0));
                    context.Response.Write(t +"-"+id);
                } else {
                    string ipAddress = System.Web.HttpContext.Current.Request.UserHostAddress;
                    int idCole = DomainModel.SesionGestion.getInfo(token, ipAddress);
                    DomainModel.Colegio cole = new DomainModel.Colegio(idCole);
                    context.Response.ContentType = "image/jpg";
                    byte[] d = (byte[])cole.getFoto(tipo, id);
                    context.Response.OutputStream.Write(d, 0, d.Length);
                }
            }
```

El procedimiento es el mismo, identificamos el token de sesión y extraemos los demás parámetros. Podemos ver que ahora los parámetros no vienen en la cabecera de la función ya que no es un servicio web, sino que hay que extraerlos de la petición HTTP. Utilizamos el mismo handler para cargar y para descargar imágenes, asi que en el parámetro 'action' definimos si queremos subir una imagen o descargarla. La imagen en si está en la petición HTTP, y accedemos a ella mediante Request.Files.Get(0).

En sentido inverso, cuando queremos descargar una imagen del servidor, debemos serializarla. Creamos un array de bytes y escribimos en la response como un OutputStream. Este mecanismo nos permite no tener las imágenes de los participantes en el sistema de ficheros del servidor, que sería comprometer su privacidad, y guardar las imágenes como binarios en la base de datos, que es más segura. Así, la única manera de ver una imagen es a través de este handler, y necesitas haber iniciado sesión y tener acceso a ese participante.

6.3 Clase del modelo

Una vez recibida una petición por el servicio web, y comprobado que los parámetros son válidos, invocamos a las clases del modelo, que contienen la lógica de la aplicación. Veamos la clase participante y la función de añadir un participante a un equipo existente, perteneciente a la clase Equipo.

```
namespace Olimpiadas.Core.DomainModel
{
    public class Participante : Persona
    {
        private int dorsal;
        private int idEquipo;
        private string dni;
        private string tsanitaria;
        private string observaciones;
        private Adulto padre;
        private Adulto madre;

        public Participante(String nombre, String apellido, String Ape2, int dors, string dni, string ts, string obs)
            : base(nombre, apellido, Ape2)
        {
            this.dorsal = dors;
            this.idEquipo = 0;

            this.dni = dni;
            tsanitaria = ts;
            observaciones = obs;
            padre = new Adulto("", "", "");
            madre = new Adulto("", "", "");
        }
```

Podemos ver que la clase participante hereda de la clase persona, y que tiene una serie de atributos privados, entre ellos 2 del tipo Adulto, que también hereda de persona.

```
internal Participante addParticipante(string nombre, string apellidos, DateTime fech
{
    Participante p = new Participante(nombre, apellidos, "", dorsal, dni, ts, obs);
    p.setEquipo(id);
    p.sexo = sexo;
    p.fechaNacimiento = fecha;
    p.Save();
    Asignar(p);
    return p;
}
```

Creamos una nueva instancia de persona, le añadimos algunos atributos no obligatorios, guardamos y asignamos un equipo, para devolver un participante válido, existente en la base de datos y con equipo asignado.

Un aspecto importante, responsabilidad de esta capa de software, es la serialización a objetos JSON, que nos permita enviar la información a los clientes. El framework .NET tiene un mecanismo propio de serialización eficiente que consiste en declarar qué atributos de la clase son serializables. En nuestro caso, para poder personalizar más la serialización, hemos implementado las funciones que devuelven un string que cumple con la arquitectura json. Veamos el ejemplo de serialización de un participante.

```
internal string SerializeSelf()
{
    // Datos basicos
    StringBuilder result = new StringBuilder("{\"Jugador\": {\"id\": \"" + getId() +
        "\",\"equipo\": \"" + idEquipo + "\",\"nombre\": \"" + nombre +
        "\", \"apellidos\": \"" + apellido1 + " " + apellido2 + "\",");
    result.Append(" \"fecha\": \"" + this.fechaNacimiento.ToString("d") +
        "\", \"sexo\": \"" + this.sexo + "\", \"status\": \"" + getStatus() + "\",");
    // equipo
    result.Append(" \"dorsal\": \"" + dorsal + "\", \"dni\": \"" + dni +
        "\", \"ts\": \"" + tsanitaria + "\", \"observaciones\": \"" + observaciones + "\",");
    // familia
    result.Append(" \"padre\": " + padre.SerializeSelf() + ", \"madre\": " + madre.SerializeSelf());

    result.Append("}}");
    return result.ToString();
}
```

Con este mecanismo podemos serializar todos los datos de un colegio con una sola llamada, y cada clase se encarga de serializar su parte. Veamos el ejemplo de un json, obteniendo con un equipo de prueba:

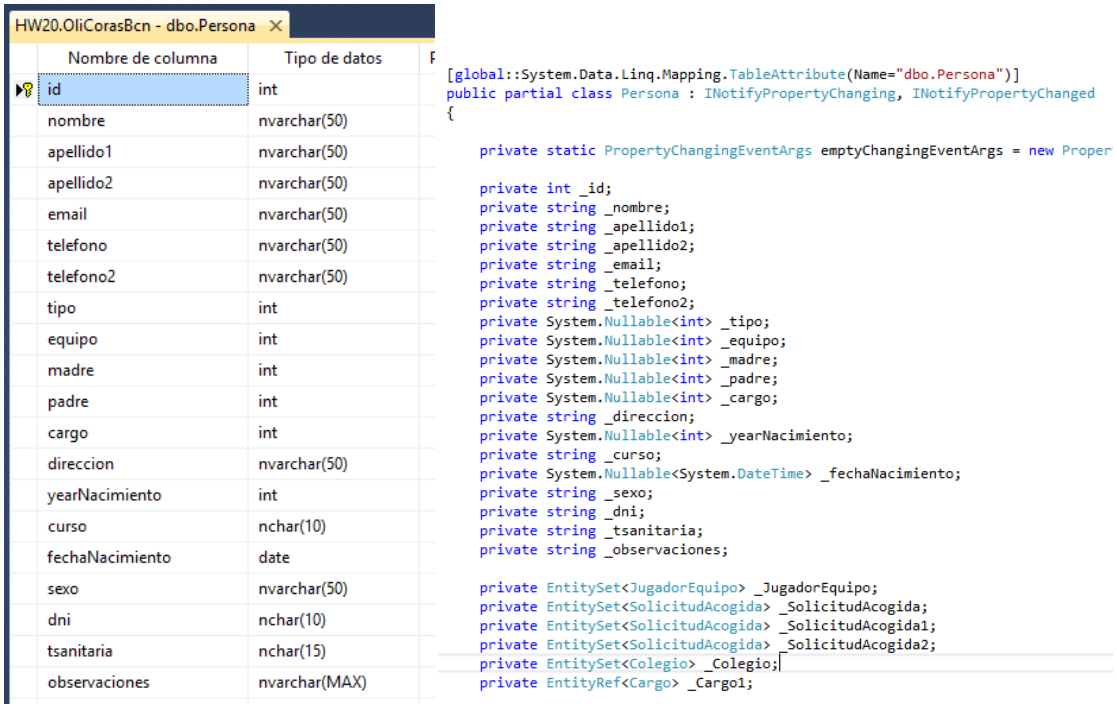

```
{
    "Equipo": {
        "id": "26",
        "nombre": "Test Team",
        "deporteid": "2",
        "deporte": "Baloncesto",
        "status": "2",
        "camprin": "Rojo",
        "pantprin": "Rojo",
        "camsec": "Azul",
        "pantsec": "Azul",
        "jugadores": [{
            "Jugador": {
                "id": "1357",
                "equipo": "26",
                "nombre": "Prueba",
                "apellidos": "Prueba",
                "fecha": "01/01/2004",
                "sexo": "Hombre",
                "status": "2",
                "dorsal": "4",
                "dni": "q        ",
                "ts": "q          ",
                "observaciones": "Prueba",
                "padre": {
                    "persona": {
                        "id": "1358",
                        "nombre": "Prueba",
                        "apellido1": "Prueba",
                        "apellido2": "",
                        "fechaNacimiento": "01/01/0001 0:00:00",
                        "sexo": ""
                    },
                    "mail": "mail@me.com",
                    "direccion": "",
                    "telefono1": "Prueba",
                    "telefono2": ""
                },
                "madre": {
                    "persona": {
                        "id": "1359",
                        "nombre": "",
                        "apellido1": "",
                        "apellido2": "",
                        "fechaNacimiento": "01/01/0001 0:00:00",
                        "sexo": ""
                    },
                    "mail": "",
                    "direccion": "",
                    "telefono1": "",
                    "telefono2": ""
                }
            }
        }],
        "entrenadores": []
    }
}
```

6.4 Clase LINQ

Antes de empezar, definimos LINQ como la interfaz entre las clases del modelo y las tablas de la base de datos. Gracias a ella podemos olvidarnos de hacer consultas SQL, y trabajar con objetos nativos del lenguaje C#. Para poder utilizarlo, debemos definir una cadena de conexión a la base de datos, y generar un archivo dbml. Este archivo es la representación de las tablas de la base de datos que queremos usar, y al generarlo, crea una serie de clases internas que nos mapean la base de datos en objetos en memoria.

Como hemos visto en apartados anteriores, hemos definido en la base de datos una tabla persona con todos los posibles datos necesarios, sin embargo en las clases de dominio tenemos una clase persona con los mínimos datos comunes y varias clases que heredan de ella. Este patrón de diseño se llama *Single Table Inheritance.* [15][16][17][18]

Aquí podemos ver la tabla Persona de la base de datos, la clase LINQ autogenerada, y las clases de dominio que se desprenden.



HW20.OliCorasBcn - dbo.Persona

Nombre de columna	Tipo de datos
id	int
nombre	nvarchar(50)
apellido1	nvarchar(50)
apellido2	nvarchar(50)
email	nvarchar(50)
telefono	nvarchar(50)
telefono2	nvarchar(50)
tipo	int
equipo	int
madre	int
padre	int
cargo	int
direccion	nvarchar(50)
yearNacimiento	int
curso	nchar(10)
fechaNacimiento	date
sexo	nvarchar(50)
dni	nchar(10)
tsanitaria	nchar(15)
observaciones	nvarchar(MAX)

```
[global::System.Data.Linq.Mapping.TableAttribute(Name="dbo.Persona")]
public partial class Persona : INotifyPropertyChanging, INotifyPropertyChanged
{

    private static PropertyChangingEventArgs emptyChangingEventArgs = new Proper

    private int _id;
    private string _nombre;
    private string _apellido1;
    private string _apellido2;
    private string _email;
    private string _telefono;
    private string _telefono2;
    private System.Nullable<int> _tipo;
    private System.Nullable<int> _equipo;
    private System.Nullable<int> _madre;
    private System.Nullable<int> _padre;
    private System.Nullable<int> _cargo;
    private string _direccion;
    private System.Nullable<int> _yearNacimiento;
    private string _curso;
    private System.Nullable<System.DateTime> _fechaNacimiento;
    private string _sexo;
    private string _dni;
    private string _tsanitaria;
    private string _observaciones;

    private EntitySet<JugadorEquipo> _JugadorEquipo;
    private EntitySet<SolicitudAcogida> _SolicitudAcogida;
    private EntitySet<SolicitudAcogida> _SolicitudAcogida1;
    private EntitySet<SolicitudAcogida> _SolicitudAcogida2;
    private EntitySet<Colegio> _Colegio;
    private EntityRef<Cargo> _Cargo1;
```

Podemos ver que de la definición de la tabla (izquierda) a la clase LINQ (derecha) hay un mapeo directo de todos los campos y de las “foreign keys”. Además de los atributos privados, se han generado múltiples funciones get/set/list para acceder a los datos.

Aquí vemos la clase de dominio Persona, de la que heredan Participante y Adulto. Cada una se responsabiliza de sus atributos, aunque el objeto LINQ que necesitan consultar las 3 sea la clase dbo.Persona

```
public class Persona
{
    private int id;
    public String nombre;
    public String apellido1;
    public String apellido2;
    public DateTime fechaNacimiento;
    public String sexo;
```

```
public class Participante : Persona
{
    private int dorsal;
    private int idEquipo;
    private string dni;
    private string tsanitaria;
    private string observaciones;
    private Adulto padre;
    private Adulto madre;
```

```
public class Adulto : Persona
{
    private String telefono1;
    private String telefono2;
    private String email;
    private String direccion;
```

Como ejemplo, vamos a ver la función de búsqueda de un participante ya existente:

```
public Participante(int id)
    : base(id)
{
    this.dorsal = 0;
    this.idEquipo = 0;

    dni = "";
    tsanitaria = "";
    observaciones = "";
    using (OliDatosDataContext db = new OliDatosDataContext())
    {
        var query = (from jug in db.JugadorEquipo where jug.jugador == id select jug).SingleOrDefault();
        this.dorsal = query.dorsal.HasValue ? query.dorsal.Value : 0;
        this.idEquipo = query.equipo;

        var query2 = (from jug in db.Persona where jug.id == id select jug).SingleOrDefault();
        this.dni = query2.dni;
        this.tsanitaria = query2.tsanitaria;
        this.observaciones = query2.observaciones;

        if (query2.padre.HasValue && query2.padre.Value > 0)
            this.padre = new Adulto(db, query2.padre.Value);
        else this.padre = new Adulto("", "", "");
        if (query2.madre.HasValue && query2.madre.Value > 0)
            this.madre = new Adulto(db, query2.madre.Value);
        else this.madre = new Adulto("", "", "");
    }

}
```

Para buscar el participante por id, se instancia un nuevo participante con el parámetro id, y lo primero que se ejecuta es la instanciación de la clase padre, Persona, con el mismo parámetro. A continuación se ejecuta la sentencia LINQ `(from jug in db.JugadorEquipo where jug.jugador == id select jug).SingleOrDefault();` que devuelve un objeto del tipo dbo.Persona, y del que tenemos que extraer los datos propios del jugador, confiando que la superclase ya habrá hecho lo mismo con los datos propios de Persona.

Veamos además un ejemplo inverso, en vez de consultar datos a la base de datos, vamos a guardar un jugador que puede haber sido modificado.

```
internal void Save()
{
    base.Save();
    padre.Save();
    madre.Save();
    using (OliDatosDataContext db = new OliDatosDataContext())
    {
        var query2 = (from ju in db.Persona where ju.id == this.getId() select ju).SingleOrDefault();
        query2.dni = this.dni;
        query2.tsanitaria = this.tsanitaria;
        query2.observaciones = this.observaciones;
        query2.padre = this.padre.getId();
        query2.madre = this.madre.getId();
        db.SubmitChanges();
    }
}
```

Primero se invoca a la función Save() de la clase Persona con `base.Save();` luego guardamos los datos de padre y madre, conseguimos el objeto dbo.Persona y guardamos los atributos propios de la clase Participante. Para finalizar, aceptamos los cambios para que se guarden en la base de datos con db.SubmitChanges().

6.5 Extras

6.5.1 Fichas en pdf

Una funcionalidad que se pidió casi al final del desarrollo fue la generación de un pdf por cada participante a modo de ficha. Para automatizar este proceso, creamos un script en lenguaje tcl que ejecutaba un programa llamado wkhtmltopdf.exe, que dada una url genera un pdf con el contenido. Además cada ficha de participante debía llamarse con un código correspondiente al colegio, al deporte y al dorsal. Para evitar que se imprimieran las cabeceras innecesarias hubo que definir el modo 'print', activado si en los parámetros de la url aparece "print=si". Este modo elimina los menús, los botones, y todo lo que no debe aparecer en los pdfs. Además, solo se genera el pdf cuando la página activa el flag "Ready to print", es decir, cuando el servidor ha devuelto los datos del participante.

El script es el siguiente:

```
package require Tcl 8.6
package require json

proc getColPrefix { id } {
        switch $id {
                2 {return B; break}
                3 {return H; break}
                4 {return LM; break}
                5 {return M; break}
                6 {return ZY; break}
                7 {return SS; break}
                9 {return T; break}
                10 {return V; break}
                11 {return G; break}
        }
}

proc getDepPrefix { dep } {
        switch $dep {
                "Baloncesto" {return B; break}
                "Balonmano" {return H; break}
                default {return F; break}
        }
}
set token "48664e14-e968-4d26-8f05-45069910e594"
set jsonpath "C:/Users/Javi.JG-Blue/Desktop/Datos.json"
set fp [open $jsonpath r]
set json_data [read $fp]
close $fp
set total 276
set current 0
set datadict [dict get [::json::json2dict $json_data] "Colegios"]
set GLOBALURL "http://core.olimpiadas.corazonistasbcn.com/Gestion/Participantes.aspx"
foreach colegioraw $datadict {
        set colegio [dict get $colegioraw Colegio]
        set cole  [dict get $colegio nombre]
        puts "Cole [dict get $colegio id]  $cole "
        set CP [getColPrefix [dict get $colegio id]]
        foreach equiporaw [dict get $colegio Equipos] {
                set equipo [dict get $equiporaw Equipo]
                set deporte  [lindex [dict get $equipo deporte] 0]
                puts "    $deporte"
                set dir "E:/Coras/$cole/$deporte"
                file mkdir $dir
                set DP [getDepPrefix $deporte]
                set i 1
                foreach jugadorraw [dict get $equipo jugadores] {
                        set jugador [dict get $jugadorraw Jugador]
                        set idj [dict get $jugador id]
                        set idjPub "$CP$DP$i"
                        puts "        jugador $idj : $idjPub"
                        incr current
                        set quedan [expr $total - $current]
                        puts "Vamos por $current de $total - Quedan $quedan a 6s por individuo:"
                        puts "[expr $quedan *6] s"
                        flush stdout
                        catch {
                        exec "C:/Program Files/wkhtmltopdf/bin/wkhtmltopdf.exe"
                        --cookie sesOliGestionBcn $token --viewport-size 1280x1024
                        --window-status ready_to_print
                        $GLOBALURL?print=si&idParticipante=$idj $dir/$idjPub.pdf &
                        } msg
                        after 5500
                        incr i
                }
        }
}

puts "Total $total"
```

Este script genera un pdf por cada participante, y los ordena en carpetas por colegios y por equipos, además, a cada pdf lo nombra según el sistema de códigos especificado por la organización. Un ejemplo de un pdf generado por el script es el siguiente:

Col.legi Corazonistas
Colegio Sagrado Corazón

DATOS DEL PARTICIPANTE

Nombre	Bruno	Equipo	Barcelona Balonmano
Apellidos	****	Tipo	Jugador
Fecha de nacimiento	****	Sexo	Hombre
DNI	****	Dorsal	1
Tarjeta sanitaria	****		

Observaciones

Alérgico a las nueces e intolerante al pescado

DATOS DE CONTACTO

Nombre padre	Juan	Apellidos padre	****
Teléfono padre	****	Mail padre	**** @gmail.com
Nombre madre	Anna	Apellidos madre	****
Teléfono madre	****	Mail madre	**** @gmail.com

6.5.2 Pantallas de televisión

Otro requerimiento de última hora fue la compra e instalación de dos pantallas de televisión para instalarlas en el patio. Estas pantallas sirvieron para publicar los marcadores en directo de los partidos en juego, avisos importantes, imágenes que los usuarios publicaban en las redes sociales y video en directo a través de la herramienta Periscope. El colegio de Barcelona compró 2 televisores LG con SmartTV, y se diseñó un apartado en la web con el contenido especificado.

7. Mejoras futuras

Como cualquier proyecto, con la experiencia y el uso salen nuevas ideas para mejorar la aplicación. Ante el interés mostrado por los demás colegios por darle continuidad al proyecto y utilizar la plataforma en futuras ediciones, se listan posibles mejoras del sistema:

- En la página mostrada en las pantallas de televisión, el contenido era actualizado manualmente. Una mejora sería que, en función de la hora, se muestren los partidos en juego y los próximos partidos automáticamente.
- Una gran mejora sería la creación de cuentas de usuario, que permitan al público seguir a un equipo, a un participante, recibir avisos en función de los equipos que sigue, etc. Para ello, el framework .NET dispone de Membership, una herramienta orientada a la gestión de cuentas y que permite personalizar el contenido a cada usuario.
- Una funcionalidad que se ha echado de menos es la integración de las redes sociales en la web. Una sección de fotos por hashtag en instagram o en twitter habría permitido a los usuarios subir contenidos y ver cómo aparecen en las pantallas y en la web.

8. Conclusiones

Este proyecto ha sido muy extenso y cambiante. Conforme se acercaba la fecha se han ido introduciendo funcionalidades nuevas o retirando algunas que habían perdido interés. Durante toda la fase de inscripciones se produjeron 2 incidencias relacionadas con la introducción por error de caracteres no estándar, que al ser guardados en la base de datos hicieron que el sistema cayera. Esto me enseñó a revisar todos y cada uno de los datos que introducen los usuarios. Se corrigieron las incidencias y se añadió un filtro a todos los campos.

Por motivos de tiempo, se cancelaron los desarrollos de las aplicaciones móviles. Gracias a que el diseño de la página web se adapta a todos los dispositivos, esto no supuso una pérdida de funcionalidad.

Durante el evento, sucedió la peor incidencia que pudo haber sucedido, llovió todo el sábado, y la pista de reserva solo estaba contratada el domingo. Hubo que cancelar toda la competición del sábado, rehacer los calendarios, y el sistema de puntuación y competición, dos veces. La parte de cambiar los partidos fue sencilla, ya que, in situ, con la nueva planificación y acceso a la base de datos, se pudo cambiar uno a uno todos los partidos. La parte de modificar el algoritmo de puntuación, debido a la falta de tiempo para hacer pruebas, se decidió eliminarla y calcular las puntuaciones a mano, conforme se jugaban los partidos.

La plataforma sirvió tanto para la inscripción y gestión de todos los equipos y participantes, como para mantener a las familias informadas durante el evento. Recibimos elogios del resto de colegios, que acostumbrados al sistema tradicional de gestionar este evento, basados en inscripciones en papel escaneadas, vieron la oportunidad de utilizar la plataforma para futuras ediciones. Las familias agradecieron estar informadas tanto de los cambios de planes como de la inmediatez de los datos de los partidos.

A nivel personal he visto la complejidad de crear un sistema web robusto y eficiente. En funcionalidades como las de administración de listar a todos los participantes del evento, un diseño ineficiente, con demasiados accesos a la base de datos, implicaba una demora de 6 segundos para acceder a la lista. Esto implicó tener que corregir diferentes funciones mientras los usuarios accedían a la plataforma, y la creación de una caché, que se destruía cada vez que algún colegio modificaba o añadía un dato, y se generaba cuando se invocaba la instrucción de listar si no existía. De esta manera se redujeron los tiempos de espera de esta instrucción de 6 segundos a milésimas.

Con una simple clase, los alumnos de 4º de ESO de la asignatura de informática del colegio pudieron usar los servicios públicos (como los marcadores de los partidos) para crear webs con el contenido en directo, e introducirse en el mundo de las tecnologías de la información.

Si tuviera que volver a empezar de nuevo este proyecto, probablemente lo haría con la misma tecnología, pero buscaría mejorar partes del diseño que he visto que son mejorables.

Lo más importante, el sistema desarrollado sirvió para mejorar la organización del evento, y permitió que la organización invirtiera más tiempo en lo realmente importante crear una gran experiencia para 272 niños participantes, que disfrutaron de un fin de semana de hermandad y deporte en Barcelona.

9. Bibliografía y referencias

[1] D. Wells, «Extreme programming,» 1999 - 2013. [En línea]. Available: http://www.extremeprogramming.org/.

[2] Microsoft Inc., «ASP.NET Web forms documentation,» [En línea]. Available: http://www.asp.net/web-forms.

[3] Microsoft Inc., «Visual Studio Documentation,» [En línea]. Available: https://code.visualstudio.com/docs.

[4] Microsoft Inc., «Sql Server 2012,» [En línea]. Available: https://msdn.microsoft.com/es-es/library/hh801901%28v=ws.11%29.aspx?f=255&MSPPError=-2147217396.

[5] W3Schools, «HTML Documentation,» [En línea]. Available: http://www.w3schools.com/html/.

[6] W3Schools, «CSS Documentation,» [En línea]. Available: http://www.w3schools.com/css/.

[7] W3Schools, «Javascript Documentation,» [En línea]. Available: http://www.w3schools.com/js/.

[8] The JQuery Foundation, «Jquery documentation,» [En línea]. Available: http://api.jquery.com/.

[9] W3Schools, «AJAX Documentation,» [En línea]. Available: http://www.w3schools.com/ajax/.

[10] ECMA International, «JSON Standard definition,» October 2013. [En línea]. Available: http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf.

[11] Microsoft Inc., «Documenting an ASMX Web Server,» [En línea]. Available: https://msdn.microsoft.com/en-us/library/aa480506.aspx.

[12] Microsoft Inc., «Internet Information Service Documentation,» [En línea]. Available: http://www.iis.net/learn.

[13] Microsoft Inc., «LINQ Documentation,» [En línea]. Available: https://msdn.microsoft.com/en-us/library/mt693024.aspx.

[14] Microsoft Inc., «C# Reference manual,» [En línea]. Available: https://msdn.microsoft.com/en-us/library/kx37x362.aspx.

[15] Microsoft Inc., «Implementation of Single Table Inheritance Pattern in ORM,» [En línea]. Available: http://www.asp.net/mvc/overview/getting-started/getting-started-with-ef-using-mvc/implementing-inheritance-with-the-entity-framework-in-an-asp-net-mvc-application.

[16] E. Mayol. A. Olivé. E. Teniente. C. Gómez, Enginyería del software: Disseny I, Barcelona: Edicions UPC, 2001.

[17] E. Gamma, Design Patterns: Elements of reusable object-oriented software, Addison Wesley, 2005.

[18] M. Fowler, Patterns of enterprise application architecture, 2003.

[19] PlayOff Informàtica, «PlayOff Informàtica,» 2014. [En línea]. Available: http://www.playoffinformatica.com/descripcion/.

[20] V. M. Plaza, «Aplicación web para la gestión de un evento deportivo,» UPC Commons, 2014.

[21] W3, «HTTP Documentation,» [En línea]. Available: https://www.w3.org/Protocols/.

[22] BOE, «Ley Orgánica de Protección de Datos Personales 15/1999,» [En línea]. Available: http://www.boe.es/buscar/doc.php?id=BOE-A-1999-23750.

[23] BOE, «Ley 34/2002, de 11 de julio, de servicios de la sociedad de la información y de comercio electrónico.,» [En línea]. Available: ttps://www.boe.es/buscar/act.php?id=BOE-A-2002-13758.

[24] BOE, «Directiva 2000/31/CE del Parlamento Europeo y del Consejo, de 8 de junio de 2000, relativa a determinados aspectos jurídicos de los servicios de la sociedad de la información, en particular el comercio electrónico en el mercado interior,» [En línea]. Available: https://www.boe.es/buscar/doc.php?id=DOUE-L-2000-81295.